Беларусі дзяржаўны універсітэт

М.М. Пісарук

Патокі ў сетках

Мінск
2009

М.М. Пісарук. Патокі ў сетках. – Мінск, Беларускі дзяржаўны універсітэт, 1996, –97с.

Аўтамабільныя дарогі, чыгунка, электрычныя сеткі, сеткі камунікацый і многія іншыя фізічныя сеткі ўвайшлі ў наша паўсядзённае жыццё. Як следства, нават для неспецыяліста відавочна практычная важнасць і шырокая распаўсюджанасць сетак. Больш таго, так як фізічныя характарыстыкі сетак (напрыклад, ўмова захавання патока ў вузлах) дапускаюць вельмі простае матэматычнае апісанне, то практыкі могуць лёгка зразумець не толькі матэматычныя пастаноўкі патокавых аптымізацыйных задач, але таксама і тыя ідэі, якія ляжаць у аснове алгарытмаў іх рашэння.

Для студэнтаў і аспірантаў матэматычных і эканамічных спецыяльнасцяў універсітэтаў.

Змест

Глава 1

Карацейшыя шляхі

Дадзен арыентаваны граф $G = (V, E)$, кожнай дузе якога прыпісаны *кошт* (*даўжсыня*) $c(v, w)$. Існуюць некалькі класічных фармулёвак задач аб пошуку карацейшых шляхоў у графах:

1. **Карацейшы шлях паміж дзвумя вылучанымі вяршынямі**. У графе G трэба знайсці шлях

   $$P = \{s = v_0, v_1, \ldots, v_{k-1}, v_k = t\}$$

   ад вяршыні $s \in V$ да вяршыні $t \in V$ мінімальнага кошту (*карацейшы шлях*)

   $$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i)\,.$$

2. **Карацейшыя шляхі ад адной вылучанай вяршыні да ўсіх астатніх**. У графе G трэба знайсці карацейшыя шляхі ад вылучанай вяршыні $s \in V$ да ўсіх астатніх вяршынь графа.

3. **Карацейшыя шляхі паміж усімі парамі вяршынь**. У графе G трэва знайсці карацейшыя шляхі паміж кожнай парай вяршынь $s, t \in V$.

Метады пошука карацейшых шляхоў у графах з'яўляюцца, з аднаго боку, падпраграмамі больш складаных метадаў прыняцця аптымальных рашэнняў, а з іншага маюць разнастайныя самастойныя прымяненні. Некалькі з якіх мы разгледзім у наступным параграфе, іншыя будуць разгледжаны пазней (гл., напрыклад, параграф 1.7). Можна таксама сцвярджаць, па меншай меры ў дачыненні да камбінаторных задач, што метады дынамічнага праграмавання з'яўляюцца ні чым іншым як алгарытмамі пошуку карацейшых шляхоў на нейкіх графах.

1.1 Прыклады задач

1.1.1 Планаванне дзейнасці гандлёвага прадпрыемства

Гандлёваму прадпрыемству трэба задаволіць спрос на нейкі прадукт на працягу T перыядаў. Абазначым чакаемы попыт у перыяд t праз d_t, $t = 1, \ldots, T$. Няхай ў перыяд t закупляецца x_t адзінак прадукта $(t = 1, \ldots, T)$, $x = (x_1, \ldots, x_T)$. Тады

$$r_t(x) \stackrel{\text{def}}{=} \sum_{i=1}^{t}(x_i - d_i) \tag{1.1}$$

ёсць астача прадукта пасля перыяду t. Вектар $x \in \mathbb{R}_+^T$ называем *планам*, калі ён задавальняе наступнай умове

$$r_t(x) \geq 0, \quad t = 1, \ldots, T-1; \quad r_T(x) = 0. \tag{1.2}$$

Затраты на закупку x_t адзінак прадукта ў перыяд t азначаюцца па правілу:

$$F_t \text{sign}(x_t) + c_t x_t \,,$$

дзе F_t — фіксаваная плата за закупку партыі прадукта, а c_t — адпаведна аптовы кошт адзінкі прадукта;

Няхай w_t — затраты, звязаныя з захаваннем адзінкі прадукта ў перыяд t. Будзем лічыць, што адзінка прадукта, якая рэалізуецца ў перыяд t, захоўваецца ў сярэднім палову гэтага перыяду; таму кошт яе захавання на працягу дадзенага перыяду роўны $\frac{1}{2}w_t$. Пры такім дапушчэнні, затраты на захаванне пратукты ў перыяд t складуць велічыню

$$w_t(x_t + r_{t-1}) - \frac{1}{2}w_t d_t = w_t \sum_{i=1}^{e}(x_i - d_i) + \frac{1}{2}w_t d_t \tag{1.3}$$

Трэба знайсці план x (*аптымальны план*), які мінімізуе сумарныя выдаткі прадпрыемства па ўсім T перыядам

$$\sum_{t=1}^{T}(F_t \text{sign}(x_t) + c_t x_t + w_t \sum_{i=1}^{t} x_i) + \sum_{t=1}^{T}(w_t(\frac{1}{2}d_t - \sum_{i=1}^{t} w_i). \tag{1.4}$$

Так як у (1.4) другое складаемае не залежыць ад x, то задачу пошука аптымальнага плана можна сфармуляваць як задачу мінімізацыі ўвагнутай функцыі

$$f(x) \stackrel{\text{def}}{=} \sum_{t=1}^{T}(F_t \text{sign}(x_t) + c_t x_t + w_t \sum_{i=1}^{t} x_i) \tag{1.5}$$

на мностве планаў.

Лема 1.1 *Існуе аптымальны план x, які задавальняе наступнай умове*

$$x_t r_{t-1} = 0 \quad \textit{для } t = 2, \ldots, T. \tag{1.6}$$

Доказ. Няхай x — нейкі план, для якога ўмова (1.6) не выконваецца, t^1 — мінімальны індэкс, для якога яна парушаецца, а t^0 — максімальны індэкс строга меньшы за t^1, такі, што $x_{t^0} > 0$. Калі $c_{t^0} + \sum_{t=t^0}^{t^1-1} w_t \le c^{t^1}$, то тады $f(x') \le f(x)$ для плана $x' \stackrel{\text{def}}{=} x + x_{t^1}(e_{t^0} - e_{t^1})$. Калі ж $c_{t^0} + \sum_{t=t^0}^{t^1-1} w_t \ge c^{t^1}$, то $f(x'') \le f(x)$ для плана $x'' \stackrel{\text{def}}{=} x + r_{t^1}(x)(e_{t^1} - e_{t^0})$. Такім чынам, мы паказалі, што ад плана x можна перайсці да іншага плана (x' ці x'') без павелічэння функцыі мэты, прычым, на новым плане ўмова (1.6) парушаецца для меньшай колькасці індэксаў. □

Лема 1.1 дазваляе звесці задачу пошука аптымальнага плана да задачы пошука карацейшых шляхоў.

Тэарэма 1.1 *Задача пошука аптымальнага плана эквівалентна задачы аб карацейшым шляху ад вяршыні* 1 *да вяршыні* $T+1$ *ў графе* $G = (V, E)$, *дзе* $V \stackrel{\text{def}}{=} \{1, \ldots, T+1\}$, $E \stackrel{\text{def}}{=} \{(i,j) : i, j \in V,\ i < j\}$, *а кошт дугі* $(i,j) \in E$ *азначаецца па правілу:*

$$c(i,j) \stackrel{\text{def}}{=} F_i + \sum_{t=i}^{j-1} (c_i + w_t) d_t.$$

Доказ. Пакажам, што паміж планамі, якія задавальняюць умове (1.6), і шляхамі ў графе G ад вяршыні 1 да вяршыні $T+1$ існуе ўзаемна адназначная адпаведнасць. Няхай план x задавальняе ўмове (1.6) і $1 = i_1 < i_2 < \ldots < i_k$ ёсць індэксы яго ненулявых кампанент. Тады

$$x_{i_j} = \sum_{t=i_j}^{i_{j+1}-1} d_t\,, \quad j = 1, \ldots, k. \tag{1.7}$$

і кошт шляху $(i_1, \ldots, i_k, i_{k+1} = T+1)$ у графе G роўны $f(x)$.

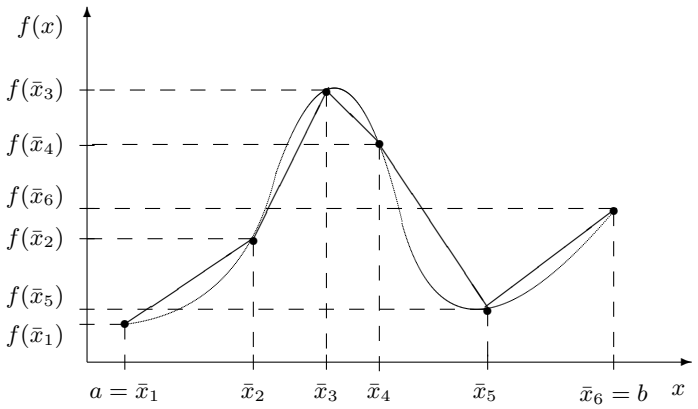
Наадварот, няхай $P = (1 = i_1, \ldots, i_k, i_{k+1} = T+1)$ ёсць шлях у графе G. Яму адпавядае план x, ненулявыя кампаненты якога азначаюцца па правілу (1.7). Няцяжка таксама ўпэўніцца, што $f(x) = c(P)$. □

1.1.2 Апраксімацыя кавалкава-лінейнай функцыі

Кавалкава-лінейныя функцыі маюць шматлікія прылажэнні. Часта такія функцыі маюць вялікую колькасць *пунктаў злому*. Прадстаўленне такіх функцый патрабуе шмат памяці і часу пры вылічэнні яе значэнняў. Часам бывае карысным апраксіраваць такую функцыю іншай кавалкава-лінейнай функцыяй з меньшай колькасцю пунктаў злому.

Няхай $f(x)$ — кавалкава-лінейная функцыя з пунктамі злому

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n). \tag{1.8}$$



Дапусцім пункты злому ўпарадкаваны такім чынам, што

$$x_1 \leq x_2 \leq \ldots \leq x_n.$$

Лічым, што n — вялікі лік і мы хочам апраксіраваць $f(x)$ іншай кавалкава-лінейнай функцыяй $\tilde{f}(x)$, якая праходзіць толькі праз нейкае падмноства кропак з (1.8), улучаючы першы і апошні пункты. У якасці прыклада разгледзім функцыю $f(x)$, якая прадстаўлена на мал. 1.1.2 суцэльнай лініяй. Яна мае дзесяць пунктаў злому. Пунктырнай лініяй прадстаўлена функцыя $\tilde{f}$, якая праходзіць толькі праз пунктаў.

Любая апраксімацыя прыводзіць да памылак пры вылічэнні значэнняў функцыі. Будзем ацэньваць памылку апраксімацыі велічынёй

$$\beta \sum_{i=1}^{n} (f(x_i) - \tilde{f}(x_i))^2$$

для нейкай канстанты β.

Мы сфармулюем праблему пошука найлепшай апраксімацыі як задачу пошука карацейшага шляху ў графе G з мноствам вяршынь $V = \{1, \ldots, n\}$ і мноствам дуг $E = \{(i,j) \,:\, i,j \in V\,,\ i < j\}$. Дуга (i,j) азначае, што мы будзем апраксіраваць функцыю $f(x)$ на адрэзку $[x_i, x_j]$ функцыяй

$$\tilde{f}(x) = f_1(x_i) + \frac{f(x_j) - f(x_i)}{x_j - x_i}(x - x_i).$$

Кошт $c(i,j)$ дугі (i,j) азначаецца па правілу:

$$c(i,j) \stackrel{\text{def}}{=} \alpha + \beta \sum_{i=1}^{n} (f(x_i) - \tilde{f}(x_i))^2,$$

дзе α ёсць "кошт"захоўвання адной кропкі.

Кожны шлях з вяршыні 1 у вяршыню n у графе G адпавядае кавалкава-лінейнай функцыі $\tilde{f}(x)$ і кошт гэтага шляху роўны агульнаму кошту захоўвання гэтай функцыі і штрафу за памылкі апраксімацыі. Зразумела, што карацейшаму шляху адпавядае аптымальная апраксімацыя.

1.1.3 Кантроль за якасцю прадукцыі

На канвееры, на якім вырабляецца нейкі прадукт, выконкаецца n аперацый. Пасля кожнай аперацыі магчымы кантроль якасці. Прадукт паступае на канвеер партыямі па $B \geq 1$ штук у партыі. Пасля кожнай аперацыі могуць выявіцца дэфекты, якія нельга выправіць. Таму дэфектныя экзэмпляры выкідваюцца ў адыходы. Няхай p_i — верагоднасць дэфекта на i-й стадыі. Пасля шэрагу аперацый (якіх канкрэтна трэба будзе вызначыць) праводзіцца кантроль якасці ўсёй прадукцыі (выбаркавы кантроль не праводзіцца). Будзем лічыць, што кантроль выяўляе ўсе дэфектныя экзэмпляры. Трэба знайсці аптымальны план кантроля за якасцю, які вызначае пасля якіх аперацый праводзіць кантроль. Меньшая колькасць інспекцый памяншае затраты на іх правядзенне, але павялічвае вытворчыя затраты, так як пры гэтым могуць выконвацца непатрэбныя аперацыі на дэфектных экзэмплярах.

Дапусцім, што дадзены наступныя параметры задачы:

(a) α_i кошт выканання i-й аперацыі на адным экзэмпляры;

(b) f_{ij} — фіксаваны кошт кантроля адной партыі прадукта пасля аперацыі j, пры ўмове, што папярэдні кантроль праводзіўся пасля аперацыі i;

(c) g_{ij} — кошт кантроля адзінкі прадукта пасля аперацыі j, пры ўмове, што папярэдні кантроль праводзіўся пасля аперацыі i;

Кошт кантроля пасля аперацыі j +залежыць ад таго, калі апошні раз ажыццяўляўся кантроль. Калі гэта было пасля аперацыі i, то кантралёр павінен выявіць усе дэфекты, якія былі зроблены на ўсіх прамежкавых этапах $i+1, i+2, \ldots, j$.

Мы сфармулюем задачу выбара аптымальнага плана кантроля як задачу аб карацейшым шляху ў графе G з мноствам вяршынь $V \stackrel{\text{def}}{=} \{0, 1, \ldots, n\}$ і мноствам дуг $E \stackrel{\text{def}}{=} \{(i,j) : i,j \in V,\ i < j\}$. Кожны шлях у графе G з вяршыні 0 да вяршыні n адпавядае плану кантроля. Напрыклад, калі $n = 6$, то шляху $(0, 2, 4, 6)$ адпавядае план, па якім кантроль ажыццяўляецца пасля 2-й, 4-й і 6-й аперацый. Азначым кошт дугі (i, j) у графе G роўным

$$c(i,j) \stackrel{\text{def}}{=} f_{i,j} + B(i)g_{ij} + B(i)\sum_{k=i+1}^{j} \alpha_k\,, \tag{1.9}$$

дзе $B(i) = B\prod_{k=1}^{i}(1-p_k)$ ёсць чакаемая колькасць недэфектных адзінак прадукта пасля аперацыі i. Першыя два складаемых у (1.9) задаюць кошт кантроля пасля ітэрацыі i, а трэці член сумы — гэта вытворчыя выдаткі.

1.2 Дрэва карацейшых шляхоў

Як гэта не парадаксальна, але знайсці карацейшы шлях паміж дзвумя вылучанымі вяршынямі не лягчэй, чым шукаць карацейшыя шляхі ад адной вяршыні да ўсіх астатніх.

Няхай $G = (V, E)$ ёсць арграф з вылучанай вяршыняй $s \in V$, Кожнай дузе $(v, w) \in E$ прыпісаны кошт $c(v, w)$. Без абмежавання агульнасці будзем лічыць, што з s *дасягаюцца* (існуе шлях ва) ўсе астатнія вяршыні графа. Калі гэта не так, то можам дабавіць да G дугі (s, v), $v \in V \setminus s$, бясконцага кошту $c(s, v) = \infty$. Кошт карацейшага шляху у графе G з вяршыні s у вяршыню $v \in V$ абазначым праз $\sigma(s, v)$ (калі такога шляху не існуе, то $\sigma(s, v) = +\infty$).

Няхай $P = (s = v_0, v_1 \ldots, v_k = v)$ ёсць карацейшы шлях ад s да v. Тады для любога $0 \leq i \leq k$ кошт падшляху $P_i = (s = v_0, v_1 \ldots, v_i)$ шляху P роўны $\sigma(s, v_i)$, бо інакш адрэзак P_i шляху P можна замяніць карацейшым шляхам ад s да v і атрымаць больш кароткі шлях P' ад s да v. Гэта ёсць *прынцып аптымальнасці*. Калі ў графе G няма цыклаў адмоўнага кошту, то ўсе карацейшыя шляхі з'яўляюцца простымі і з прынцыпа аптымальнасці можна заключыць, што кошты карацейшых шляхоў задавальняюць наступным ураўненням Бэлмана:

$$\begin{aligned} \sigma(s, s) &= 0, \\ \sigma(s, v) &= \min_{(w,v)\in E} (\sigma(s, w) + c(w, v)) \quad \text{для ўсіх } v \in V \setminus s. \end{aligned} \tag{1.10}$$

Наступныя два паняцці, якія ідуць ад лінейнага праграмавання, адыгрываюць фундаментальную ролю ва ўсёй патокавай аптымізацыі. *Функцыя цэн* (*вектар патынцыялаў*) ёсць функцыя $p : V \to \mathbb{R}$. *Прыведзеная функцыя кошту адносна функцыі цэн* p азначаецца па правілу:

$$c_p(v, w) = c(v, w) + p(v) - p(w).$$

Цэны вяршынь маюць натуральную эканамічную інтэрпрэтацыю як дзеючыя рынкавыя цэны на нейкі прадукт. Мы можам інтэрпрэтаваць прыведзены кошт $c_p(v, w)$, як суму затрат на закупку адзінкі прадукта ў вяршыні v па цане $p(v)$ і затрат на транспартыроўку ў w мінус даход ад продажу яе там па цане $p(w)$.

Лема 1.2 *Няхай $G = (V, E)$ ёсць арграф, на дугах якога азначана функцыя кошту c, а на вяршынях функцыя цэн p. Тады для шляху P з вяршыні v у вяршыню w у графе G мае месца роўнасць $c_p(P) = c(P) + p(v) - p(w)$. У прыватным выпадку, калі P — цыкл, $c_p(P) = c(P)$.*

Доказ. Няхай $P = (v = v_0, v_1, \ldots, v_{k-1}, v_k = w)$. Тады

$$\begin{aligned} c_p(P) &= \sum_{i=1}^{k} c_p(v_{i-1}, v_i) \\ &= \sum_{i=1}^{k} (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i)) \\ &= \sum_{i=1}^{k} c(v_{i-1}, v_i) + \sum_{i=1}^{k} p(v_{i-1}) - \sum_{i=1}^{k} p(v_i) \\ &= c(P) + p(v) - p(w). \end{aligned}$$

□

Для пакрывальнага ардрэва T з корнем s *функцыя адлегласцей* $d : V \to \mathbb{R}$ азначаецца рэкурсіўна наступным чынам:

$$d(s) = 0,\ d(v) = d(parent(v)) + c(parent(v), v) \quad \text{для } v \in V \setminus s\,,$$

дзе $parent(v)$ ёсць *бацька* (пачатковая вяршыня адзінай дугі, якая ўваходзіць у v) вяршыні v у дрэве T. Пакрывальнае ардрэва T з корнем s называецца *дрэвам карацейшых шляхоў*, калі для кожнай вяршыні v адзіны шлях дрэва T з s у v з'яўляецца карацейшым шляхам з s у v у графе G, г. зн., $d(v) = \sigma(s, v)$.

Тэарэма 1.2 *Няхай усе вяршыні графа $G = (V, E)$ дасягаюцца з вяршыні $s \in V$. Граф G мае дрэва карацейшых шляхоў тады і толькі тады, калі ён не мае цыклаў адмоўнага кошту. Пакрывальнае ардрэва T з корнем s з'яўляецца дрэвам карацейшых шляхоў тады і толькі тады, калі яго функцыя адлегласцей d задавальняе ўмове*

$$c_d(v, w) \geq 0 \quad \textit{для ўсіх } (v, w) \in E. \tag{1.11}$$

Доказ. Калі G не мае цыклаў адмоўнага кошту, то карацейшыя шляхі ад s да ўсіх астатніх вяршынь з'яўляюцца простымі. Няхай D ёсць аб'яднанне дуг гэтых шляхоў. Відавочна, што граф (V, D) утрымлівае пакрывальнае ардрэва T, якое з'яўляецца дрэвам карацейшых шляхоў.

Зразумела, што функцыя адлегласцей дрэва карацейшых шляхоў задавальняе ўмове (1.11). Дакажам адваротнае. Няхай дрэва T задавальняе ўмове (1.11). Па леме 1.2 для любога цыкла Γ графа G маем $c(\Gamma) = c_d(\Gamma) \geq 0$. Няхай $s = v_0, v_1, \ldots, v_k = v$ ёсць карацейшы шлях з s у v. Так як G не мае цыклаў адмоўнага кошту, то кошт карацейшага шляху з s у s роўны $0 = d(s)$. Па індукцыі дапусцім, што $d(v_{k-1})$ ёсць кошт карацейшага шляху з s у v_{k-1}. Так як па (1.11) $d(v) \leq d(v_{k-1}) + c(v_{k-1}, v)$, а $d(v_{k-1}) + c(v_{k-1}, v)$ ёсць кошт карацейшаго шляху з s у v, то $d(v)$ — таксама кошт карацейшага шляху з s у v. □

Вынік 1.1 *Арграф $G(V,E)$ не мае цыклаў адмоўнага кошту тады і толькі тады, калі існуе функцыя $d : V \to \mathbb{R}$, якая задавальняе ўмове (1.11).*

Доказ. **Дастатковасць** умовы (1.11) сцвярджаецца ў леме 1.2. Дакажам **неабходнасць**. Пабудуем дапаможны арграф $G_{aux} = (V_{aux}, E_{aux}) = (V \cup \{s\}, E \cup (\{s\} \times V))$, дадаючы да графа G новую вяршыню s і мноства дуг, якія выходзяць з s ва ўсе астатнія вяршыні. Кошты новых дуг азначым роўнымі нулю. Відавочна, калі G не мае цыклаў адмоўнага кошту, то іх няма і ў G_{aux}. Таму па тэарэме 1.2 граф G_{aux} мае дрэва карацейшых шляхоў і яго функцыя адлегласцей d задавальняе ўмове (1.11). □

1.3 Алгарытм пабудовы дрэва карацейшых шляхоў

Ідэя ўсіх алгарытмаў пошука карацейшых шляхоў у графах аднолькавая. Усе яны пачынаюць з нейкай функцыі $d : V \to \mathbb{R}$, такой, што $d(s) = 0$ і $d(v) \geq \sigma(s,v)$ для ўсіх $v \in V$. Затым ітэратыўна паўтараецца наступны крок:

> выбраць дугу (v,w) адмоўнага прыведзенага кошту $c_d(v,w) < 0$ і замяніць $d(w)$ на $d(v) + c(v,w)$.

Гэта ёсць *метад паслядоўнай апраксімацыі*. Калі ў графе няма цыклаў адмоўнага кошту, то ў выпадку, калі кошты дуг цэлалікавыя, пасля канцай колькасці ітэрацый метад спрыняецца. Пры гэтым $d(v) = \sigma(s,v)$ для ўсіх $v \in V$. Пасля гэтага карацейшы шлях ад s да дадзенай вяршыні v можна знайсці працэдурай *find_path*, якая прыведзена на мал. 1.1.

```
find_path(G, d, s, v, path) // d(v) ≠ ∞
{
  for (; v ≠ s; v := x) {
    v → path;
    (x, v) ∈ arg{d(w) + c(w, v) = d(v) : (w, v) ∈ E(V, v)};
  }
}
```

Малюнак 1.1: Працэдура *find_path*

1.3.1 Алгарытм Форда-Бэлмана

Эфектыўнасць метада паслядоўнай апраксімацыі істотна залежыць ад парадку выбара дуг адмоўнага прыведзенага кошту. У гэты параграфе мы будзем разглядаць алгарытм, які прапанаваны незалежна Фордам і Бэлманам. Працэдура *shortest_path*, якая рэалізуе гэты алгарытм, прадстаўлена

на мал. 1.2. Калі працэдура вяртае значэнне **true**, то ўказальнікі *parent* задаюць дрэва карацейшых шляхоў, а функцыя d ёсць яго функцыя адлегласцей. Калі ж працэдура вяртае значэнне **false**, то любы цыкл падграфа $(V, \overline{E})$, дзе

$$\overline{E} \stackrel{\text{def}}{=} \{(parent(v), v) \,:\, v \in V,\ parent(v) \neq \textbf{nil}\}\,,$$

з'яўляецца цыклам адмоўнага кошту ў графе G. Каб гарантаваць, што ўсе вяршыні графа G дасягаюцца з s, калі $(s, v) \notin E$, мы дабаўляем (умоўна) гэтую дугу да G і прыпісваем ёй кошт ∞. Калі пасля завяршэння алгарытма $d(v) = \infty$, то у графе G вяршыня v недасягаецца з s.

Абазначым праз $\sigma^i(s, v)$ кошт карацейшага шляху з s у v сярод усіх шляхоў, якія маюць роўна i дуг. Калі такога шляху не існуе, то $\sigma^i(s, v) = \infty$. Няхай $d^i(v)$, S^i — адпаведна $d(v)$ і спіс S пасля ітэрацыі i. Этап ініцыялізацыі называем 0-й ітэрацыяй.

```
shortest_path(G, c, s, parent, d, S)
{
  for (v ∈ V \ s) { d(v) := ∞; parent(v) :=nil;}
  d(s) := 0; S := {s};
  for (i := 1; i ≤ n; i := i + 1) { // галоўны цыкл
    Q := ∅; dw := d;
    for ((v, w) ∈ E(S, V))
      if (d(w) > dw(v) + c(v, w)) {
        d(w) := dw(v) + c(v, w);
        parent(w) := v; Q ← w;
      }
    if ((S := Q) = ∅) return true;
  }
  return false;
}
```

Малюнак 1.2: Алгарытм Форда-Бэлмана пошука карацейшых шляхоў

Лема 1.3 *Выконваюцца наступныя роўнасці*

a) $d^i(v) = \min_{0 \le k \le i} \sigma^k(s, v)$ *для ўсіх* $v \in V$,

b) $d^i(v) = \sigma^i(s, v) < \sigma^{i-1}(s, v)$ *для ўсіх* $v \in S^i$,

Доказ. Відавочна, што сцвярджэнне лемы справядліва пасля 0-й ітэрацыі. Дапусцім, што пасля заканчэння $(i-1)$-й (перад пачаткам i-й) ітэрацыі алгарытма

$$\begin{aligned} d^{i-1}(v) &= \min_{0 \le k \le i-1} \sigma^k(s, v) \quad \text{для ўсіх } v \in V\,, \\ d^{i-1}(v) &= \sigma^{i-1}(s, v) \quad \text{для ўсіх } v \in S^{i-1}. \end{aligned}$$

Няхай $w \in V$ і $s = v_0, v_1, \ldots, v_{i-1}, v_i = w$ ёсць шлях кошту $\sigma^i(s,w)$ у графе G. Тады па дапушчэнню $d^{i-1}(v_{i-1}) = \sigma^{i-1}(v_{i-1})$. Таму $v_{i-1} \in S^{i-1}$ і, калі $d^{i-1}(w) > d^{i-1}(v_{i-1}) + c(v_{i-1}, w) = \sigma^i(s,w)$, то пасля i-й ітэрацыі $d^i(w) = \sigma^i(s,w)$, а w будзе ўлучана ў S_i. □

Лема 1.4 *Калі пасля завяршэння працедуры shortest_path $S = \emptyset$, то функцыя адлегласцей d задавальняе ўмове (1.11).*

Доказ. Заўважым, што $d^i(v) \geq d^{i+1}(v)$ для ўсіх $v \in V$. Няхай $(v,w) \in E$. Так як $S = \emptyset$, то $d(v) = d^i(v)$ для нейкага $1 \leq i \leq n-1$. Таму

$$d(w) \leq d^{i+1}(w) \leq d^i(v) + c(v,w) = d(v) + c(v,w).$$

□

Лема 1.5 *Граф G мае цыкл адмоўнага кошту тады і толькі тады, калі пасля завяршэння працэдуры shortest_path мноства S не пустое.*

Доказ. Калі $S = \emptyset$, то па тэарэме 1.2 і леме 1.4 граф G не мае цыклаў адмоўнага кошту. Калі $S \neq \emptyset$, то па леме 1.3 для $v \in S$

$$d^n(v) = \sigma^n(s,v) < \sigma^i(s,v) \quad \text{для ўсіх } 0 \leq i < n.$$

Відавочна, што кожны шлях з s у v даўжыні n і кошту $\sigma^n(s,v)$ абыходзіць цыкл адмоўнага кошту. □

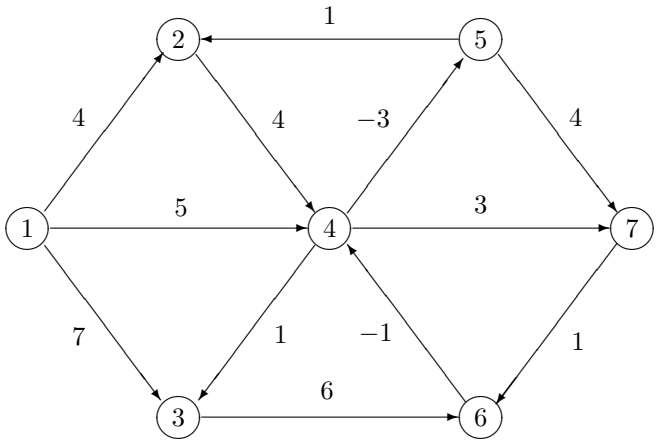
Тэарэма 1.3 *За час $O(nm)$ працэдура shortest_path або будуе дрэва карацейшых шляхоў, або знаходзіць цыкл адмоўнага кошту.*

Доказ. Карэктнасць алгарытма вынікае з лем 1.4 і 1.5 Так як складанасць адной ітэрацыі працэдуры *shortest_path* не пераўзыходзіць $O(m)$, колькасць ітэрацый не больш за n, складанасць працэдуры *find_cycle* — $O(n)$, то агульная складанасць працэдуры *shortest_path* ёсць $O(nm)$. □

Прыклад 1.1 *У графе, які прадстаўлены на мал. 1.3, знайсці карацейшыя шляхі ад вяршыні* 1 *да ўсіх астатніх вяршынь.*

Праца алгарытма па ітэрацыях прадстаўлена ў табл. 1.1. Этап ініцыялізацыі ёсць ітэрацыя 0. Функцыя dw — гэта функцыя d на папярэдняй ітэрацыі, а мноства Q — гэта мноства S на наступнай ітэрацыі. Дрэва карацейшых шляхоў задаецца функцыяй *parent* на апошняй 5-й ітэрацыі. Яно таксама прадстаўлена на мал. 1.4. □

Заўвага 1.1 *Пры апісанні алгарытма Форда-Бэлмана мы ўвялі функцыю dw толькі для таго, каб атрымаць суадносіны a) і b) з лемы 1.3. Зразумела, што складанасць алгарытма не пагоршыцца, калі замест dw выкарыстоўваць функцыю d. Наадварот, у сярэднім алгарытм будзе працаваць хутчэй.*



Малюнак 1.3: Граф да прыкладу 1.1

Табліца 1.1:

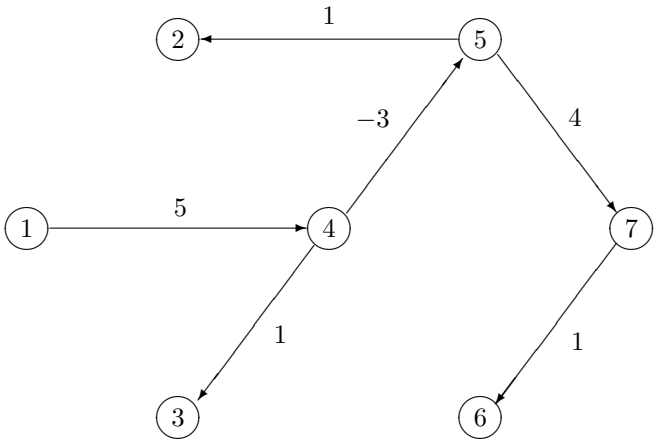
Іт.	S	d	*parent*
0	$\{1\}$	$(0, \infty, \infty, \infty, \infty, \infty, \infty)$	(**nil**, **nil**, **nil**, **nil**, **nil**, **nil**, **nil**)
1	$\{2, 3, 4\}$	$(0, 4, 7, 5, \infty, \infty, \infty)$	(**nil**, 1, 1, 1, **nil**, **nil**, **nil**)
2	$\{3, 5, 6, 7\}$	$(0, 4, 6, 5, 2, 13, 8)$	(**nil**, 1, 4, 1, 4, 3, 4)
3	$\{2, 6, 7\}$	$(0, 3, 6, 5, 2, 9, 6)$	(**nil**, 5, 4, 1, 4, 7, 5)
4	$\{6\}$	$(0, 3, 6, 5, 2, 7, 6)$	(**nil**, 5, 4, 1, 4, 7, 5)
5	$\emptyset$	$(0, 3, 6, 5, 2, 7, 6)$	(**nil**, 5, 4, 1, 4, 7, 5)

1.3.2 Алгарытм Дэйкстры

Існуе шэраг сітуацый, у якіх задача пошуку карацейшых шляхоў рашаецца асабліва проста. Адна з такіх сітуацый, калі кошты ўсіх дуг неадмоўныя. У гэтым выпадку для пабудовы дрэва карацейшых шляхоў існуе больш эфектыўны алгарытм Дэйкстры, які прыведзены на мал. 1.5. На кожнай стадыі алгарытм падзяляе мноства вяршынь V на два падмноствы: S і $V \setminus S$. Калі вяршыня v належыць S, то карацейшы шлях да яе ўжо знойдзены і роўны $d(v)$. На чарговай ітэрацыі алгарытм выбірае вяршыню $w \in V \setminus S$ з мінімальнай "меткай"$d(w)$, дадае яе да S і для ўсіх дуг $(w, v) \in E(w, V \setminus S$ пералічвае меткі канцавых вяршынь па правілу:

$$d(v) = \min\{d(v), d(w) + c(w, v)\}.$$

Лема 1.6 *Алгарытм Дэйкстры падтрымлівае наступныя інварыянты:*



Малюнак 1.4: Дрэва карацейшых шляхоў да прыкладу 1.1

a) $d(v) \leq d(w)$ *для ўсіх* $v \in S$, $w \in V \setminus S$;

b) $d(v) + c(v, w) \geq d(w)$ *для ўсіх* $(v, w) \in E(V, S)$.

Доказ. Індукцыяй па $|S|$. Дэталі пакідаем чытачу. □

Тэарэма 1.4 *Калі кошты ўсіх дуг неадмоўныя, то за час* $O(n^2)$ *алгарытм Дэйкстры будуе дрэва карацейшых шляхоў.*

Доказ. Карэктнасць алгарытма вынікае з лемы 1.6. Ацэнім складанасць алгарытма. На этапе ініцыялізацыі патрабуецца $O(n)$ аперацый. Складанасць адной ітэрацыі $O(n)$. А так як колькасць усіх ітэрацый роўна $n - 1$, то складанасць усяго алгарытма $O(n^2)$. □

Прыклад 1.2 *У графе, прадстаўленым на мал. 1.6, трэба знайсці карацейшыя шляхі ад вяршыні 1 да ўсіх астатніх вяршынь*

Праца алгарытма па ітэрацыях прадстаўлена ў табл. **??**, а дрэва карацейшых шляхоў адлюстравана на мал. 1.7.

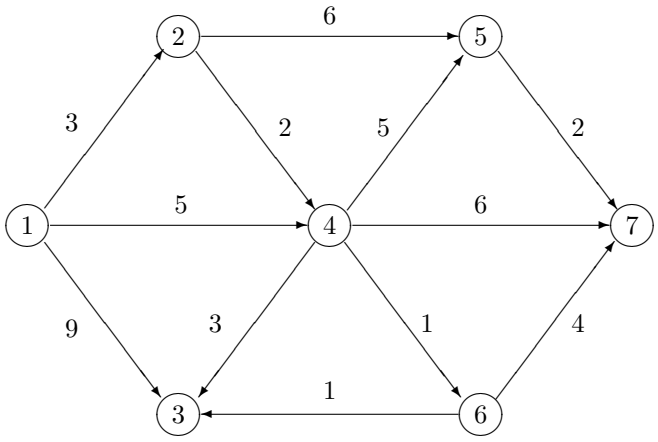
□

1.4 Карацейшыя шляхі з абмежаваннем на час праезду

Няхай $V = (V, E)$ — арыентаваны граф, кожнай дузе (v, w) якога прыпісаны два лікі: *кошт праезду* $c(v, w)$ і *час праезду* $t(v, w) \geq 0$. Нам трэба знайсці

```
nonnegative_cost_shortest_path(G, c, s, parent, d)
{
  for (v ∈ V) {d(v) := ∞; parent(v):=nil;}
  d(s) := 0; S := ∅;
  for (;|S| < n − 1;) { // галоўны цыкл
    w ∈ arg min{d(v) : v ∉ S};
    S := S ∪ {w};
    for ((w, v) ∈ E(w, V \ S))
        if (d(v) > d(w) + c(w, v)) {
          d(v) := d(w) + c(w, v); parent(v) := w;
        }
  }
}
```

Малюнак 1.5: Алгарытм Дэйкстры



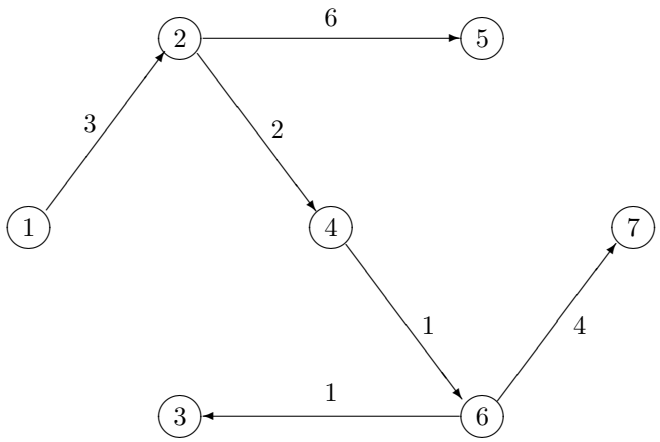
Малюнак 1.6: Граф да прыкладу 1.2

шляхі мінімальнага кошту ад вяршыні $s \in V$ да ўсіх вяршынь $v \in V$, пры ўмове што час праезду па іх не пераўзыходзіць T адзінак часу. Як прыклад прымянення такой задачы разгледзім сітуацыю, калі кампанія па дастаўцы пошты хоча мінімізаваць транспартныя выдаткі і таксама гарантаваць, што пошта будзе дастаўляцца не пазней чым праз T адзінак часу.

Абазначым праз $d_t(v)$ кошт карацейшага шляху з s у v сярод шляхоў, якія патрабуюць не больш чым t адзінак часу. Няцяжка ўстанавіць наступ-

Табліца 1.2:

Іт.	w	S	d	*parent*
1	1	$\{1\}$	$(0,3,9,5,\infty,\infty,\infty)$	$(\mathbf{nil},1,1,1,\mathbf{nil},\mathbf{nil},\mathbf{nil})$
2	2	$\{1,2\}$	$(0,3,9,5,9,\infty,\infty)$	$(\mathbf{nil},1,1,1,2,\mathbf{nil},\mathbf{nil})$
3	4	$\{1,2,4\}$	$(0,3,8,5,9,6,11)$	$(\mathbf{nil},1,4,1,2,4,4)$
4	6	$\{1,2,4,6\}$	$(0,3,7,5,9,6,10)$	$(\mathbf{nil},1,6,1,2,4,6)$
5	3	$\{1,2,4,6,3\}$	$(0,3,7,5,9,6,10)$	$(\mathbf{nil},1,6,1,2,4,6)$
6	5	$\{1,2,4,6,3,5\}$	$(0,3,7,5,9,6,10)$	$(\mathbf{nil},1,6,1,2,4,6)$



Малюнак 1.7: Дрэва карацейшых шляхоў да прыкладу 1.2

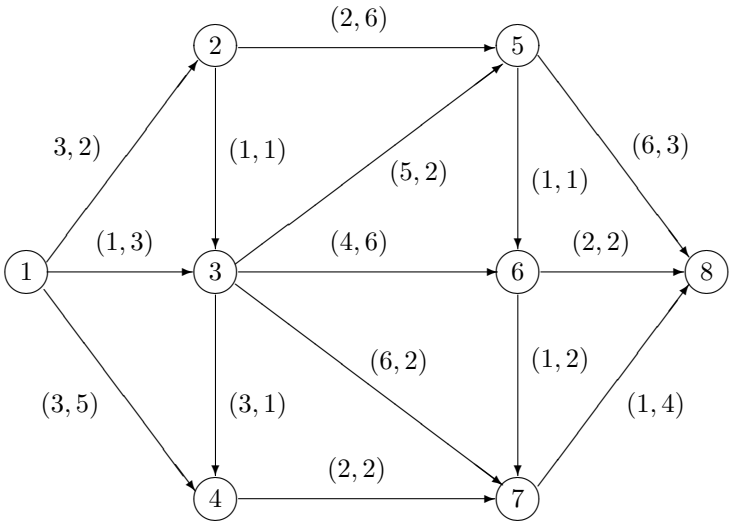
ныя рэкурэнтныя ўраўненні:

$$\begin{aligned} d_0(s) &= 0, \\ d_0(v) &= \infty, \quad v \in V \setminus \{s\}, \\ d_t(v) &= \min\{d_{t-1}(v), \min_{\substack{(w,v)\in E \\ t\geq t(w,v)}} d_{t-t(w,v)}(w) + c(w,v)\}. \end{aligned} \tag{1.12}$$

Ураўненні (1.12) можна разглядаць як абагульненне ўраўненняў Бэлмана (1.10). Няцяжка пераканацца, што для падліку ўсіх значэнняў $d_t(v)$ ($t = 0,\ldots,T$; $v \in V$) патрэбна выканаць $O(n^2T)$ арыфметычных аперацыі.

Прыклад 1.3 *Вырашыць задачу пошука карацейшых шляхоў ад вяршыні $s = 1$ з абмежаваннем на час праезду $T = 10$ для графа, які прадстаўлены на мал. 1.8.*

Велічыні $d_t(v)$ прадстаўлены ў табл. 1.3.



Малюнак 1.8: Граф да прыкладу 1.3

Табліца 1.3:

t	0	1	2	3	4	5	6	7	8	9	10
$d_t(1)$	0	0	0	0	0	0	0	0	0	0	0
$d_t(2)$	∞	∞	3	3	3	3	3	3	3	3	3
$d_t(3)$	∞	∞	∞	1	1	1	1	1	1	1	1
$d_t(4)$	∞	∞	∞	∞	4	3	3	3	3	3	3
$d_t(5)$	∞	∞	∞	∞	∞	6	6	6	5	5	5
$d_t(6)$	∞	∞	∞	∞	∞	∞	7	7	7	5	5
$d_t(7)$	∞	∞	∞	∞	∞	7	6	5	5	5	5
$d_t(8)$	∞	∞	∞	∞	∞	∞	∞	∞	9	8	7

Адваротным ходам знойдзем аптымальны шлях ад вяршыні 1 да вяршыні 8.

$$\begin{aligned} d_{10}(8) &= \min\{d_9(8), d_7(5) + c(5,8), d_8(6) + c(6,8), d_6(7) + c(7,8)\} \\ &= \min\{8, 12, 9, 7\} = 7. \end{aligned}$$

Таму ў вяршыню 8 трэба ісці з вяршыні 7. Вылічваем

$$\begin{aligned} d_6(7) &= \min\{d_5(7), d_4(3) + c(3,7), d_4(4) + c(4,7), d_4(6) + c(6,7)\} \\ &= \min\{7, 7, 6, \infty\} = 7 \end{aligned}$$

і заключаем, што ў вяршыню 7 трэба ісці з вяршыні 4. Так як

$$\begin{aligned} d_4(4) &= \min\{d_3(4), d_3(3) + c(3,4)\} \\ &= \min\{\infty, 4\} = 4, \end{aligned}$$

то ў вяршыню 4 трэба ісці з вяршыні 3. Паколькі

$$\begin{aligned} d_3(3) &= \min\{d_2(3), d_0(1) + c(1,3), d_2(2) + c(2,3)\} \\ &= \min\{\infty, 3, 4\} = 3, \end{aligned}$$

то ў вяршыню 3 трэба ісці з вяршыні 1. Таму $(1, 3, 4, 7, 8)$ — аптымальны шлях з вяршыні 1 у вяршыню 8. □

1.5 Карацейшыя шляхі паміж усімі парамі вяршынь

Дадзен граф $G = (V, E)$ і функцыя $c : E \to \mathbb{R}$ коштаў яго дуг. Трэба знайсці карацейшыя шляхі паміж усімі парамі вяршынь $v, w \in V$ графа G. Зразумела, што гэтую задачу можна вырашыць, калі знайсці карацейшыя шляхі ад кожнай вяршыні $s \in V$ з дапамогай алгарытма Форда-Бэлмана, выканаўшы ў суме $O(n^2m)$ арыфметычных аперацый. Для прадстаўлення адказа спатрэбіцца $O(n^2)$ ячэек памяці для захоўвання n функцый *parent*. Ніжэй мы прыводзім крыху больш эфектыўны варыянт такога падыходу да рашэння задачы.

1. Спосабам, указаным у доказе выніка 1.1, алгарытмам Форда-Бэлмана знаходзім функцыю цэн p, такую, што $c_p(v, w) \geq 0$ для ўсіх $(v, w) \in E$, ці даказваем, што такой функцыі не існуе. У апошнім выпадку граф G мае цыкл адмоўнага кошту і таму задача не мае рашэння. Інакш пераходзім да кроку 2.

2. У графе G з функцыяй кошту c_p для кожнай вяршыні $s \in V$ алгарытмам Дэйкстры будуем дрэва карацейшых шляхоў (функцыю *parent*) ад s да ўсіх астатніх вяршынь.

Карэктнасць замены функцыі кошту c функцыяй прыведзеных коштаў c_p на кроку 2 вынікае з лемы 1.2. Складанасць алгарытма $O(n^3)$ ($O(nm)$ на кроку 1 і $O(n^3)$ на кроку 2).

На практыцы разгледжаны намі вышэй алгарытм прымяняюць, калі трэба знайсці карацейшыя шляхі паміж некалькімі (але не паміж усімі) парамі вяршынь графа. У тых выпадках, калі патрэбна ведаць карацейшыя шляхі паміж усімі парамі вяршынь, прымяняюць, выдатны па сваёй прастаце *алгарытм Флойда-Ўоршэла*. Алгарытм працуе з матрыцай $[d_{ij}]$ памеру $n \times n$, элементы якой спачатку роўны коштам дуг графа G, г. зн. $d_{vw} = c(v, w)$, калі $(v, w) \in E$, і $d_{vw} = \infty$, калі $(v, w) \notin E$. Ядром алгарытма з'яўляецца

```
all_pairs_shortest_paths(n, d, θ )
{
 for (i := 1; i ≤ n; i := i + 1) for (j := 1; j ≤ n; j := j + 1) θ_{i,j} := 0;
 for (k := 1; k ≤ n; k := k + 1)
    for (i := 1; i ≤ n; i := i + 1) if (i ≠ k)
       for (j := 1; j ≤ n; j := j + 1) if (j ≠ k)
          if (d_ij > d_ik + d_kj) {
           d_ij := d_ik + d_kj; θ_ij := k;
           if (i = j and d_ii < 0)
              return; // выяўлен цыкл адмоўнага кошту
          }
}

find_path(i, j)
{
 if (θ_ij = 0) return (i, j);
 else return find_path(i, θ_ij) & find_path(θ_ij, j);
}
```

Малюнак 1.9: Алгарытм Флойда-Ўоршэла

аперацыя трохвугольніка, якая для фіксаванага k $(1 \le k \le n)$ пералічвае матрыцу $[d_{ij}]$ па правілу:

$$d_{ij} := \min\{d_{ij}, d_{ik} + d_{kj}\} \quad \text{для ўсіх } i, j = 1, \ldots, n;\ i, j \ne k.$$

Для прадстаўлення карацейшых шляхоў выкарыстоўваецца матрыца $[\theta_{ij}]_{n\times n}$, дзе θ_{ij} ёсць максімальны нумар прамежкавай вяршыні, якая ляжыць на шляху з i у j. Спачатку $\theta_{ij} = 0$ для ўсіх $i, j = 1, \ldots, n$. Працэдура *all_shortest_path*, якая рэалізуе алгарытм Флойда-Ўоршэла, прыведзена на мал. 1.9. Там жа прыведзена працэдура *find_path*, якая для дзвюх дадзеных вяршынь i і j вяртае карацейшы шлях з i у j.

Вывучыўшы выкладзены вышэй матэрыял, вы ў стане самастойна даказаць наступную тэарэму.

Тэарэма 1.5 *За час $O(n^3)$ алгарытм Флойда-Ўоршэла або будуе матрыцу $[d_{ij}]$ карацейшых шляхоў паміж усімі парамі вяршынь, або знаходзіць цыкл адмоўнага кошту.*

1.6 Карацейшыя шляхі ў ацыклічным арграфе

У гэтым пункце мы будзем разглядаць задачу пошуку карацейшых шляхоў у ацыклічным арграфе. Кошты дуг могуць быць адвольныя: як дадатныя, так і адмоўныя. Па тэарэме 1.2 гэтая задача мае рашэнне. Калі граф

G не мае арыентаваных цыклаў, то з дапамогай працэдуры *topsort* можна правесці яго тапалагічную сартыроўку, г.зн., знайсці такую нумарацыю $label : V \to \{1, \ldots, n\}$ яго вяршынь, пры якой для кожнай дугі $(v, w) \in E$ выконваецца $label(v) < label(w)$. Працэдура *acyclic_shortest_path*, якая для кожнай вяршыні v вылічвае кошт $d(v)$ карацейшага шляху ад s да v, прадстаўлена на мал. 1.10. Карэктнасць рэкурэнтнай формулы

$$d(v) := \min\{d(tail(e)) + c(e) : e \in in\_edge(v)\}, \tag{1.13}$$

выкарыстанай ў алгарытме, вельмі проста даказваецца па індукцыі. А так як алгарытм разглядае кожную дугу графа роўна адзін раз, то справядліва

Тэарэма 1.6 *Працэдура* acyclic_shortest_path *знаходзіць кошты карацейшых шляхоў ад вяршыні s да ўсіх астатніх вяршынь за час $O(\max\{n, m\})$.*

```
acyclic_shortest_path(V, c, label, in_edge, d)
{
  d(s) := 0;
  for (i = 2; i <= n; ++i) {
    v := label^{-1}(i);
    d(v) := min{d(tail(e)) + c(e) : e ∈ in_edge(v)};
  }
}
```

Малюнак 1.10: Працэдура пошуку карацейшых шляхоў ў ацыклічным графе

Нагадаем, што граф $G = (V, E)$ называецца *k–долевым*, калі мноства яго вяршынь ёсць аб'яднанне неперасякальных мностваў $V_1, \ldots, V_k$ і для кожнай дугі $(v, w) \in E$ існуе i $(1 \le i \le k-1)$, такое, што $v \in V_i, w \in V_{i+1}$. Няхай $E_i = E \cap (V_i \times V_{i+1})$.

Зразумела, што k-долевы граф з'яўляецца ацыклічным, прычым усякая нумарыцыя яго вяршынь, пры якой вяршыня з долі i мае меньшы нумар, чым усякая вяршыня долі $i+1$, з'яўляецца тапалагічнай сартыроўкай. Будзем лічыць, што $s \in E$. Тады працэдуру *acyclic_shortest_path* можна запісаць у выглядзе, які прадстаўлены на мал. 1.11.

1.7 Сеткавае планаванне

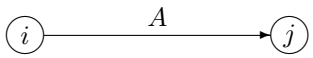
Кіраванне буйнымі праектамі звязана з вырашэннем складаных праблем планавання, вызначэння тэрмінаў пачатку і заканчэння асобных работ, кантролю за выкананнем гэтых тэрмінаў. Усё гэта ўскладняецца тым, што работы павінны выконвацца ў дадзенай тэхналагічнай паслядоўнасці.

```
k_fold_shortest_path(V, E, c, k, d)
{
  d(s) := 0;
  for (v ∈ V_1 \ s)
      d(v) := ∞;
  for (i = 2; i <= k; ++i)
        for (v ∈ V_i)
          d(v) := min{d(w) + c(w, v) : (w, v) ∈ E};
}
```

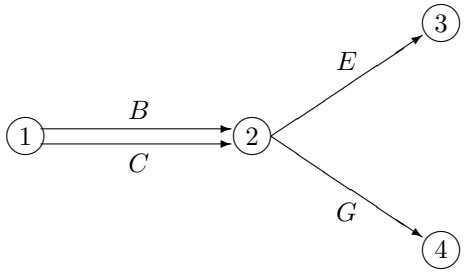
Малюнак 1.11: Працэдура пошуку карацейшых шляхоў ў k–долевым графе

1.7.1 Сеткавая мадэль праекта — сеткавы графік

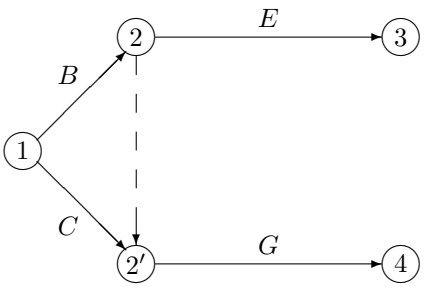
Сеткавы графік ёсць арграф $G = (V, E)$, дугі якога адпавядаюць работам, а сам граф адлюстроўвае сувязі паміж усімі заданнямі, неабходнымі для заканчэння праекту. Дугі графа з'яўляюцца *работамі*, а вяршыні — падзеямі. *Падзея* — момант часу, калі можна пачаць выкананне новых работ. Для кожнай работы (дугі) (i, j) вядома яе працягласць, г. зн. час, які патрэбны на яе выкананне. Напрамак дуг вызначаецца *суадносінамі папярэднічання*. На адрэзку сеткі



i-я падзея павінна адбыцца да пачатку работы A, а j-я падзея не можа адбыцца да заканчэння работы A. Часам дачыненні папярэднічання паміж работамі нельга дакладна адлюстраваць з дапамогай сеткі. Напрыклад, калі работа G выконваецца за работамі B і C, а работа E — за работай B, але не за C.



Прыведзенае адлюстраванне папярэднічання работ памылковае, бо з яго вынікае, што работа E ідзе і за работай C, а гэтага не патрабавалася ў зыходных ўмовах. Для правільнага прадстаўлення, трэбна ўвесці *фіктыўную* работу працягласці 0. Фіктыўныя работы будзем маляваць пунктырам.



Прыклад 1.4 *Пры зборцы нейкага варштата вузлы 1 і 2 аб'ядноўваюцца ў вузел 4, а спалучэнне вузлоў 3 і 4 дае гатовы выраб. Так як неабходна ўзгадніць нейкія дэталі вузла 3 з адпаведнымі дэталямі вузла 2, то вузел 3 немагчыма сабраць раней, чым будуць ў наяўнасці дэталі вузла 2. Асноўныя работы прыведзены ў табл. 1.4.*

Табліца 1.4: Спіс работ да прыкладу 1.4

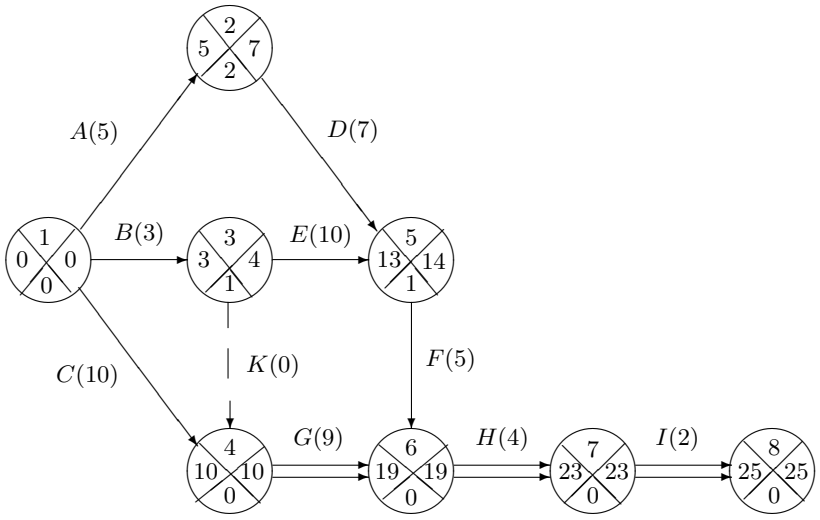
Работа		Прац.	Непаср.
Абазн.	Апісанне	(сут.)	папяр.
A	Набыццё дэталяў вузла 1	5	–
B	Набыццё дэталяў вузла 2	3	–
C	Набыццё дэталяў вузла 3	10	–
D	Выраб вузла 1	7	A
E	Выраб вузла 2	10	B
F	Выраб вузла 4	5	D, E
G	Выраб вузла 3	9	B, C
H	Канчатковая зборка	4	F, G
I	Выпрабаванні	2	H

Сеткавы графік працэсу вырабу варштата прадстаўлены на мал. 1.12.
□

Сеткавы графік праекта задавальняе наступным умовам.

1. Маецца адна пачатковая падзея (вяршыня, у якую не ўваходзіць ніводнае дугі) і адна *заключная* падзея (вяршыня, з якой не выходзіць ніводная дуга).

2. У графіку няма цыклаў. Таму падзеі можна занумараваць такім чынам, што кожная дуга (работа) пачынаецца ў вяршыні з меншым нумарам і заканчваецца ў вяршыні з большым нумарам. У далейшым будзем лічыць, што $V = \{1, \ldots, n\}$ і, калі $(i, j) \in E$, то $i < j$.



Малюнак 1.12: Сеткавы графік працэсу вырабу варштата

Нумарацыю вяршынь можна выканаць, напрыклад, такім чынам: знаходзім вяршыню, ў якую не ўваходзіць ніводная дуга і прыпісваем ёй нумар 1. Мысленна выдаляем гэтую вяршыню і інцыдэнтныя ёй дугі з графіка. У застаўшымся падграфе зноў знаходзім вяршыню, ў якую не ўваходзіць ніводная дуга і прыпісваем ёй нумар 2. Працэс паўтараем, пакуль усе вяршыні не атрымаюць нумары. Такая нумарацыя вяршынь ацыклічнага графа называецца *тапалагічнай сартыроўкай*.

1.7.2 Метад крытычнага шляху

Адной з галоўных мэт сеткавага планавання з'яўляецца атрыманне інфармацыі аб планавых тэрмінах выканання асобных работ.

Раннія і познія тэрміны надыходу падзей

Ранні тэрмін T_j^E надыходу падзеі j ёсць ранні тэрмін заканчэння ўсіх работ, якія ляжаць на шляху з першай падзеі ў j-ю падзею. Такім чынам, T_j^E ёсць кошт максімальнага шляху з вяршыні 1 у вяршыню j, калі за кошт дугі ўзяць працягласць выканання работ. Параметры T_j^E можна вылічыць па наступнай рэкурэнтнай формуле:

$$\begin{aligned} T_1^E &= 0, \\ T_j^E &= \max_{(i,j)\in E}\{T_i^E + t_{ij}\}, \quad j = 2, \ldots, n. \end{aligned}$$

У сеткавым графіку на мал. 1.12 раннія тэрміны надыходу падзей прадстаўлены ў левых сектарах.

Ранні тэрмін надыходу заключнай падзеі n роўны раняму тэрміну выканання ўсяго праекта. Такім чынам, ранні тэрмін заканчэння ўсяго праекта роўны даўжыні максімальнага шляху з пачатковай падзеі да заключнай падзеі. Гэты шлях называецца *крытычны шляхам*, а яго даўжыня — *крытычным часам*, якую абазначаем T^{kr}. На мал. 1.12 дугі крытычнага шляху намаляваны дзвумя стрэлкамі.

Позні тэрмін T_j^L надыходу падзеі j — гэта найбольш позні тэрмін надыходу падзеі j, які не ўплывае на ранні тэрмін заканчэння ўсяго праекта ў цэлым (крытычны час). Каб не парушыць ранняга тэрміну завяршэння праекта, j-я падзея павінна надысці ў момант

$$T_j^L = T^{kr} - L_{jn},$$

дзе L_{jn} — кошт максімальнага шляху з j у n. Мы можам вылічыць параметры T_j^L па наступнай рэкурэнтнай формуле:

$$\begin{aligned} T_n^L &= T^{kr}, \\ T_j^L &= \min_{(j,i)\in E}\{T_i^E - t_{ij}\}, \quad j = n-1, \ldots, 1. \end{aligned}$$

У сеткавым графіку на мал. 1.12 познія тэрміны надыходу падзей прадстаўлены ў правых сектарах.

Рэзерв часу R_j падзеі j — гэта максімальны час, на які можа быць затрыманы надыход падзеі без павелічэння ранняга тэрміну заканчэння праекта, г.зн.,

$$R_j = T_j^L - T_j^E.$$

Падзея з нулявым рэзервам часу знаходзіцца на крытычным шляху. Затрымка надыходу любой падзеі на крытычным шляху прыводзіць да затрымкі ўсяго праекта. Наадварот, падзеі, якія не ляжаць на крытычным шляху, могуць быць затрыманы на час, меншы або роўны R_j, і гэта не прывядзе да павялічэння тэрміну заканчэння ўсяго праекта. На мал. 1.12 рэзервы часу падзей прадстаўлены ў ніжніх сектарах.

Раннія і познія тэрміны пачатку і заканчэння работ

1. *Ранні тэрмін $T_B^E(i,j)$ пачатку работы* (i,j) роўны раняму тэрміну T_i^E надыходу падзеі i, паколькі работа (i,j) не можа быць распачата, пакуль не надыдзе падзея i.

2. *Позні тэрмін $T_F^L(i,j)$ заканчэння работы* (i,j) — гэта найпазнейшы тэрмін заканчэння работы (i,j) без затрымкі тэрміну заканчэння праекта: $T_B^E(i,j) = T_j^L$.

3. *Ранні тэрмін $T_F^E(i,j)$ заканчэння работы* (i,j) вызначаецца формулай $T_F^E(i,j) = T_j^E + t_{ij}$.

3. *Позні тэрмін* $T_B^L(i,j)$ *пачатку работы* (i,j) вызначаецца формулай $T_B^L(i,j) = T_j^L - t_{ij}$.

Чатыры паказчыкі рэзерву часу работы

Гэтыя паказчыкі могуць быць выкарыстаны кіраўніком праекта пры размеркаванні рэсурсаў (напрыклад, працоўных) для выканання асобных работ праекта, бо працягласць работы залежыць ад колькасці выдзеленых рэсурсаў.

1. *Сумарны рэзерв* $R^{Sum}(i,j)$ часу работы (i,j) — гэта максімальная затрымка работы (i,j) без затрымкі тэрміну ажыццяўлення ўсяго праекта:

$$R^{Sum}(i,j) = T_j^L - T_i^E - t_{ij}.$$

Для работ на крытычным шляху $R^{Sum} = 0$. Калі работа (i,j) цалкам выкарыстоўвае сумарны рэзерв, то ў з'явіцца новы крытычны шлях, які праходзіць праз дугу (i,j).

2. *Свабодны рэзерв* $R^{Free}(i,j)$ часу работы (i,j) — гэта максімальная затрымка работы (i,j), якая не ўплывае на пачатак наступных работ, г. зн., што наступныя работы могуць пачынацца ў свае раннія тэрміны:

$$R^{Free}(i,j) = T_j^E - T_i^E - t_{ij}.$$

3. *Гарантаваны рэзерв* $R^{Gar}(i,j)$ часу работы (i,j) — гэта максімальная магчымая затрымка работы (i,j), якая не ўплывае на тэрмін заканчэння ўсяго праекта пры ўмове, што папярэднія работы выконваліся са спазненнем, г.зн., заканчваліся ў свае познія тэрміны.

$$R^{Gar}(i,j) = T_j^L - (T_i^L + c_{ij})$$

4. *Незалежны рэзерв* $R^{Ind}(i,j)$ часу работы (i,j) — гэта такая затрымка работы (i,j), якая не ўплывае на пачатак наступных работ, пры ўмове, што ўсе папярэднія работы скончыліся ў свае познія тэрміны:

$$R^{Ind}(i,j) = \max\{0, T_j^E - T_i^L - t_{ij}\}$$

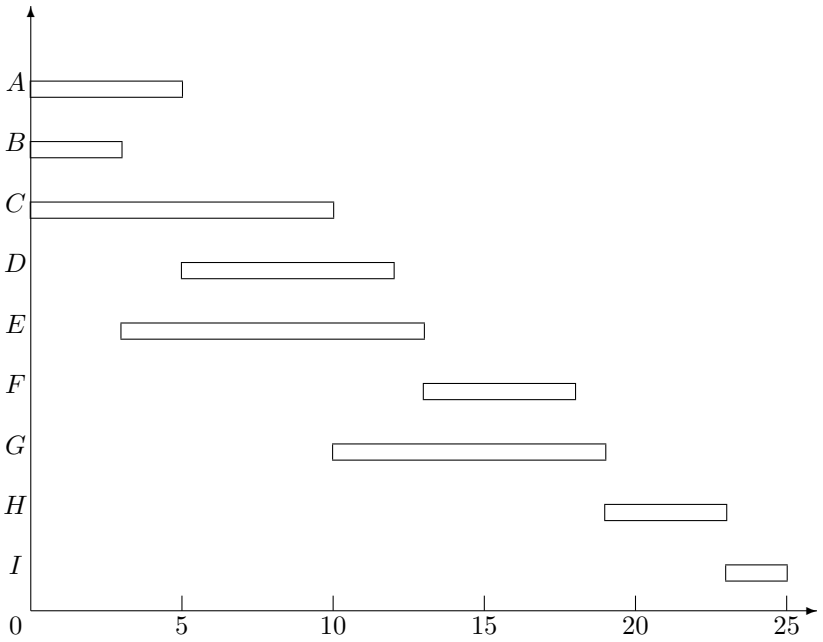
Працяг прыкладу 1.4. Вынікі працы метада крытычнага шляху прадстаўлены ў табл. 1.5. У апошнім слупку гэтай табліцы рэзервы часу работ прадстаўлены ў наступным парадку: $(R^{Sum}, R^{Free}, R^{Ind}, R^{Gar})$. □

Часавая дыяграма праекта

Можна намаляваць часавую дыяграму праекта: па восі O_x — час, які прайшоў з пачатковага моманту, па восі O_y — работы (часавая дыяграма для прыкладу 1.4 прыведзена на мал. 1.13).

Табліца 1.5:

Раб.	Дуга	Прац.	Ранні тэрмін		Позні тэрмін		Рэзервы
			Пач.	Зак.	Пач.	Зак.	
A	$(1,2)$	5	0	5	2	7	$(2,0,0,2)$
B	$(1,3)$	3	0	3	1	4	$(1,0,0,4)$
C	$(1,4)$	10	0	10	0	10	$(0,0,0,0)$
D	$(1,5)$	7	5	12	7	14	$(2,1,0,0)$
	$(3,4)$	0	3	3	10	10	$(7,7,0,0)$
E	$(3,5)$	10	3	13	4	14	$(1,0,0,0)$
G	$(4,6)$	9	10	19	10	19	$(0,0,0,0)$
F	$(5,6)$	5	13	19	14	19	$(1,1,0,0)$
H	$(6,7)$	4	19	23	19	23	$(0,0,0,0)$
I	$(7,8)$	2	23	25	23	25	$(0,0,0,0)$



Малюнак 1.13: Часавая дыяграма праекту да прыкладу 1.4

1.8 Цыклы мінімальнага сярэдняга кошту

Дадзены граф $G = (V, E)$ і функцыя $c : E \to \mathbb{R}$ коштаў яго дуг. Па аналогіі з задачай аб карацейшым шляху можна сфармуляваць задачу аб *мінімальным цыкле*, у якой у графе G трэба знайсці (просты) цыкл Y мінімальнага кошту $c(Y)$. Калі кошты ўсіх дуг роўны -1, то мінімальным цыклам будзе цыкл з максімальнай колькасцю дуг. Таму любы алгарытм для рашэння задачы аб мінімальным цыкле здольны правяраць, ці з'яўляецца дадзены граф *гамільтонавым*. А гэта азначае, што задача аб мінімальным цыкле з'яўляецца **NP**-цяжкай. Як мы ўбачым ніжэй, значна больш простай з'яўляецца задача аб мінімальным сярэднім цыкле.

Сярэднім коштам цыкла Y арграфа G называецца дачыненне яго кошту да колькасці яго дуг (ці вяршынь) $\frac{c(Y)}{|Y|}$. У задачы аб *мінімальным сярэднім цыкле*

$$\mu(G, c) \stackrel{\text{def}}{=} \min\{c(Y)/|Y| \,:\, Y \text{ — цыкл } G\}$$

трэба знайсці цыкл графа G, які мае мінімальны сярэдні кошт. Лік $\mu(G, c)$ называецца *мінімальным сярэднім цыклічным коштам графа* G *адносна* c.

1.8.1 Алгарытм Карпа

Калі арграф G не з'яўляецца моцна звязным, то мы можам знайсці мінімальны сярэдні кошт цыклаў у кожнай яго моцна звязнай кампаненце і выбраць сярод іх мінімальны. Таму будзем лічыць, што арграф G з'яўляецца моцна звязным.

Выбярэм адвольную вяршыню $s \in V$.

Лема 1.7 *Калі* $\mu(G, c) = 0$, *то*

$$\min_{v \in V} \max_{0 \le k \le n-1} \left[\frac{\sigma^n(s, v) - \sigma^k(s, v)}{n - k} \right] = 0.$$

Доказ. Так як $\mu(G, c) = 0$, то G мае цыкл нулявога кошту і не мае цыклаў адмоўнага кошту. Таму колькасць дуг у шляху мінімальнага кошту з s у t не пераўзыходзіць $n - 1$. Няхай $d(v)$ ёсць кошт гэтага шляху. Тады $\sigma^n(s, v) \ge d(v)$. Акрамя таго

$$\begin{aligned} d(v) &= \min_{0 \le k \le n-1} \sigma^k(s, v), \\ \sigma^n(s, v) - d(v) &= \max_{0 \le k \le n-1} (\sigma^n(s, v) - \sigma^k(s, v). \end{aligned}$$

Адкуль маем

$$\max_{0 \le k \le n-1} \left[\frac{\sigma^n(s, v) - \sigma^k(s, v)}{n - k} \right] \ge 0. \tag{1.14}$$

няроўнасць (1.14) пераўтвараецца ў роўнасць тады і толькі тады, калі $\sigma^n(s, v) = d(v)$. Адсюль вынікае, што мы можам завяршыць доказ, калі пакажам, што маецца вяршыня v, для якой $\sigma^n(s, v) = d(v)$. Няхай Y ёсць цыкл нулявога

кошту, і няхай w — вяршыня на гэтым цыкле. Няхай, далей, $P(w)$ — шлях кошту $d(v)$ з s у w. Тады $P(w)$ з наступным абыходам цыкла Y любую колькасць разоў таксама будзе шляхам P' мінімальнага кошту з s у w. Можна лічыць, што колькасць дуг гэтага шляху большая чым n. Няхай P'' ёсць пачатковая частка шляху P' даўжыні n, а v — канчаткавая вяршыня шляху P''. Тады P'' ёсць шлях мінімальнага кошту з s у v, г. зн., $\sigma^n(s,v) = d(v)$. □

Тэарэма 1.7 *Мае месца наступная роўнасць*

$$\mu(G,c) = \min_{v\in V} \max_{0\le k\le n-1} \left[\frac{\sigma^n(s,v) - \sigma^k(s,v)}{n-k}\right]. \tag{1.15}$$

Доказ. Калі мы адымем канстанту a ад кошту кожнай дугі $e \in E$, то, відавочна, $\mu(G,c)$ паменьшыцца на a, $\sigma^k(s,v)$ — на ka, а $(\sigma^n(s,v) - \sigma^k(s,v))/(n-k)$ — на a. Таму абедзве часткі (1.15) зменьшацца аднолькава. Няхай $a = -\mu(G,c)$. Тады $\mu(G,c^a) = 0$, дзе $c^a(v,w) \stackrel{\text{def}}{=} c(v,w) + a$, і тэарэма вынікае з лемы 1.7. □

Алгарытм Карпа

1. Вылічваем велічыні $\{\sigma^k(s,v)\}$ па рэкурэнтнай формуле

$$\sigma^k(s,v) = \min_{(w,v)\in E} \left(\sigma^{k-1}(s,w) + c(w,v)\right), \quad k = 1,\dots,n$$

   з пачатковымі значэннямі

$$\sigma^0(s,s) = 0, \quad \sigma^0(s,v) = \infty, \quad v \in V \setminus s.$$

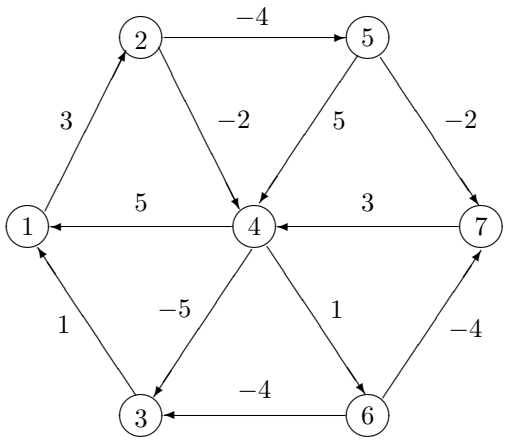
2. Знаходзім вяршыню v і лік k, на якіх дасягаецца значэнне $\mu(G,c)$.

3. Адваротным ходам знаходзім карацейшы шлях з s у v даўжыні n і вядзяляем у ім цыкл з $n-k$ дуг.

Тэарэма 1.8 *Калі граф G моцна звязны, за час $O(nm)$, выкарыстоўваючы $O(n^2)$ ячэек памяці, алгарытм Карпа знаходзіць цыкл мінімальнага сярэдняга кошту.*

Доказ. Карэктнасць алгарытма вынікае з тэарэмы 1.7. Ацэнку складанасці мы прапануем чытачам зрабіць самастойна. □

Прыклад 1.5 *Знайсці цыкл мінімальнага сярэдняга кошту ў графе, які прадстаўлены на мал. 1.14.*



Малюнак 1.14: Граф да прыкладу 1.5

Табліца 1.6:

v	1	2	3	4	5	6	7
σ^0	0	∞	∞	∞	∞	∞	∞
σ^1	∞	3	∞	∞	∞	∞	∞
σ^2	∞	∞	∞	1	-1	∞	∞
σ^3	6	∞	-4	4	∞	2	-3
σ^4	-3	9	-2	0	∞	5	-2
σ^5	-1	0	-5	1	5	1	1
σ^6	-4	2	-4	-2	-4	2	-3
σ^7	-3	-1	-7	0	-2	-1	-6
μ	1	$-\frac{1}{2}$	$-\frac{3}{4}$	2	2	$-\frac{3}{4}$	$-\frac{3}{4}$
k	6	5	3	6	6	3	3

Рэзультаты прамога ходу алгарытма прадстаўлены ў табл. 1.6. З перадапошняга радка табл. 1.6, у якім прадстаўлены лікі

$$\mu(v) \stackrel{\text{def}}{=} \max_{0\le k\le n-1}\left[\frac{\sigma^n(s,v)-\sigma^k(s,v)}{n-k}\right],$$

мы бачым, што

$$\mu(G,c)=\min_{v\in V}\mu(v)=-\frac{3}{4},$$

і гэтае значэнне дасягаецца пры $v=3$ і $k=3$. Таму цыкл мінімальнага сярэдняга кошту мае $n-k=4$ дугі. Каб знайсці гэты цыкл выконваем

адваротны ход:

$$\begin{array}{rclrcl} \sigma^7(3) & = & \sigma^6(4)-5, & \sigma^6(4) & = & \sigma^5(2)-4, \\ \sigma^5(2) & = & \sigma^4(1)+3, & \sigma^4(1) & = & \sigma^3(3)+1. \end{array}$$

Знойдзен цыкл $Y=(3,1,2,4,3)$, сярэдні кошт якога роўны $-3/4$.

1.8.2 Двойны метад

У гэтым параграфе мы разгледзім яшчэ адзін алгарытм для пошуку мінімальных сярэдніх цыклаў. У канцы параграфа мы абмяркуем яго недахопы і перавагі ў параўнанні з метадам Карпа.

Лема 1.8 *Мае месца наступная формула*

$$\mu(G,c)=\max_{p:V\to\mathbb{R}}\min_{(v,w)\in E} c_p(v,w). \tag{1.16}$$

Доказ. Справядлівасць формулы (1.16) непасрэдна вынікае з двух наступных простых назіранняў:

- для любога сапраўднага a мае месца роўнасць $\mu(G,c^a)=\mu(G,c)+a$, дзе $c^a(v,w)=c(v,w)+a$;
- $c_p(Y)=c(Y)$ для любых цыкла Y і функцыі цэн p (лема 1.2).

Цяпер павінна быць зразумелым, што $\mu(G,c)\ge\min_{(v,w)\in E} c_p(v,w)$. Таксама зразумела, што ўсе цыклы графа G маюць неадмоўныя прыведзеныя кошты адносна функцыі кошту $c^{-\mu(G,c)}$, прычым, цыклы мінімальнага сярэдняга кошту будуць мець прыведзены кошт роўны нулю. Таму, згодна выніку 1.1, існуе функцыя цэн p, такая, што $c_p^{-\mu(G,c)}(v,w)\ge 0$ для ўсіх $(v,w)\in E$. Для гэтай функцыі цэн можна праверыць, што $\mu(G,c)=\min\{c_p(v,w):(v,w)\in E\}$. □

Няхай $p: V\to\mathbb{R}$ ёсць нейкая функцыя цэн. Мы будзем называць дугу (x,y) *мінімальнай адносна* p, калі

$$c_p(x,y)=\mu(p)\stackrel{\text{def}}{=}\min_{(v,w)\in E} c_p(v,w).$$

Граф $G(p)\stackrel{\text{def}}{=}(V,E(p))$, дзе $E(p)\stackrel{\text{def}}{=}\{(v,w)\in E: c_p(v,w)=\mu(p)\}$, назавем *мінімальным падграфам адносна* p. Функцыю цэн p назавем *аптымальнай*, калі мінімальны падграф $G(p)$ мае арцыкл. Гэтае азначэнне мы матывуем формулай (1.16), з якой вынікае, што кожны цыкл графа $G(p)$ з'яўляецца мінімальным сярэднім цыклам.

Вельмі просты алгарытм, які прадстаўлены на мал. 1.15, шукае аптымальную функцыю цэн p. На ўваход працэдуры *min_mean_cycle* акрамя графа G і коштаў дуг c падаецца функцыя цэн p (напрыклад, $p\equiv 0$). Першы цыкл, выяўлены гэтым алгарытмам, з'яўляецца цыклам мінімальнага сярэдняга кошту.

min_mean_cycle(G, c, p)
{
for (; граф $G(p)$ ацыклічны;) {
Рэкурсіўна вылічваем узроўні $l(v)$ вяршынь графа $G(p)$:
калі v не мае ўваходзячых дуг, то $l(v) := 0$,
інакш $l(v) := \max\limits_{(w,v)\in E(p)} l(w) + 1$;
$\epsilon := \min\limits_{(v,w)\in E\,:\,l(v)\geq l(w)} \dfrac{c_p(v,w) - \mu(p)}{l(v) - l(w) + 1}$;
if $(\epsilon = \infty)$ **return**; // граф G ацыклічны
$p := p - \epsilon l$;
}
}

Малюнак 1.15: Двойны метад пошуку мінімальнага сярэдняга цыкла

Тэарэма 1.9 *Калі граф G не з'яўляецца ацыклічным, алгарыт на мал. 1.15 за час $O(n^2m)$ знаходзіць цыкл мінімальнага сярэдняга кошту.*

Доказ. Няхай p і p' ёсць функцыя цэн у пачатку і ў канцы нейкай ітэрацыі.

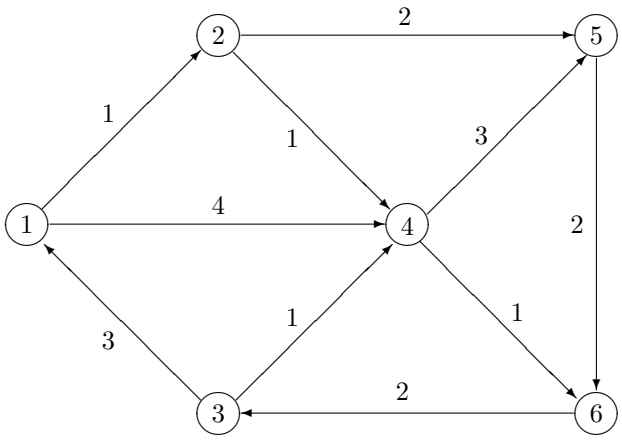
Так як граф $G(p)$ ацыклічны, то *ўзровень* $l(v)$ кожнай вяршыні $v \in V$ азначаецца карэктна і ўсе ўзроўні можна вылічыць за час $O(m)$, выкарыстоўваючы пошук у глыбіню. Працэдура пошуку ў глыбіню будуе таксама сям'ю вяршынна неперасякальных дрэваў з коранямі ў вяршынях з нулявымі ўзроўнямі. Дугі гэтых дрэваў называюць *дугамі дрэва*.

Пасля змены цэн прыведзены кошт кожнай дугі (v, w) *накіраванай уперад* $(l(v) < l(w))$ павялічваецца на $\epsilon(l(w) - l(v))$. Зазначым, што накіраванымі ўперад з'яўляюцца ўсе мінімальныя адносна p дугі. Таму новыя прыведзеныя кошты ўсіх накіраваных уперад дуг будуць не меншымі чым $\mu(p)+\epsilon$, а прыведзеныя кошты дуг дрэва дакладна роўны $\mu(p)+\epsilon$. Прыведзены кошт дугі (v, w), для якой $l(v) \geq l(w)$, памяншаецца на $\epsilon(l(w) - l(v))$. Выбарам ϵ мы гарантуем, што для ўсіх $(v, w) \in E$,

$$c_{p'}(v, w) = c_p(v, w) + \epsilon(l(w) - l(v)) \geq \mu(p) + \epsilon.$$

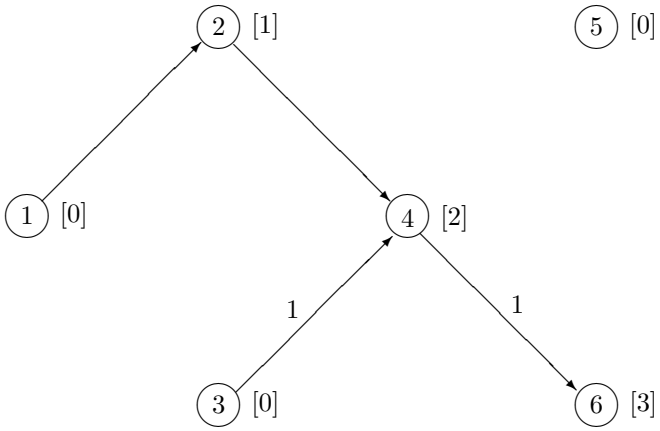
Са сказанага вышэй вынікае, што $\mu(p') = \mu(p) + \epsilon$. Паколькі ўсе дугі дрэва застаюцца мінімальнымі і адносна p', то пасля змены цэн узроўні ўсіх вяршынь не паменшацца. Больш таго, хаця б адна дуга (v, w) з $l(v) \geq l(w)$ (менавіта тая, на якой дасягаецца значэнне ϵ) стане мінімальнай адносна p'; ўзровень вяршыні w павялічыцца па меншай меры на адзінку. З гэтага мы адразу атрымліваем ацэнку $O(n^2)$ на колькасць ітэрацый, так як сума ўсіх узроўняў вяршынь не можа быць большай $(n - 1)(n - 2)/2$. □

Прыклад 1.6 *Знайсці мінімальны сярэдні цыкл у графе, які прадстаўлены на мал. 1.16.*



Малюнак 1.16: Граф да прыкладу 1.6

Пачнем рашэнне задачы двойным метадам з функцыі цэн $p^0 \equiv 0$.
1. $\mu(p^0) = 1$, а граф $G(p^0)$ прадстаўлены на мал. 1.17. Так як $l =$



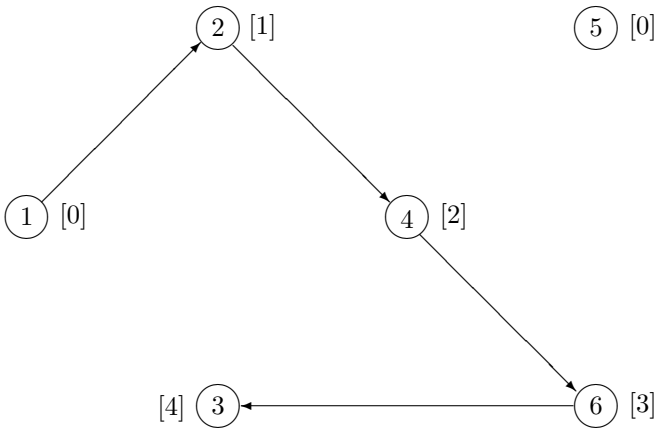
Малюнак 1.17: Граф $G(p^0)$

$(0, 1, 0, 2, 0, 3)$, то

$$\begin{aligned} \epsilon^1 &= \min\left\{\frac{c_{p^0}(2,5)-\mu(p^0)}{l(2)-l(5)+1}\frac{c_{p^0}(3,1)-\mu(p^0)}{l(3)-l(1)+1}, \frac{c_{p^0}(4,5)-\mu(p^0)}{l(4)-l(5)+1},\right. \\ &= \left.\frac{c_{p^0}(6,3)-\mu(p^0)}{l(6)-l(3)+1},\right\} = \min\left\{\frac{1}{2}, 2, 1, \frac{1}{4}\right\} = \frac{1}{4} \end{aligned}$$

і $p^1 = p^0 - \epsilon l = (0, -1/4, 0, -1/2, 0, -3/4)$.

2. $\mu(p^1) = 5/4$, а граф $G(p^1)$ прадстаўлены на мал. 1.18. Так як $l =$



Малюнак 1.18: Граф $G(p^1)$

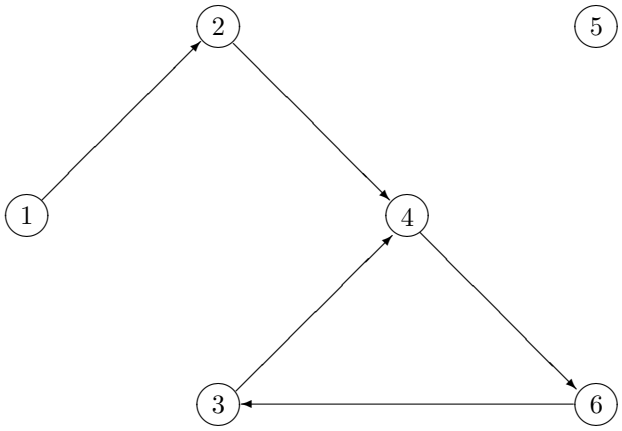
$(0, 1, 4, 2, 0, 3)$, то

$$\begin{aligned}\epsilon^2 &= \min\left\{\frac{c_{p^1}(2,5)-\mu(p^1)}{l(2)-l(5)+1}, \frac{c_{p^1}(3,1)-\mu(p^1)}{l(3)-l(1)+1},\right.\\ &\qquad \left.\frac{c_{p^1}(3,4)-\mu(p^1)}{l(3)-l(4)+1}, \frac{c_{p^1}(4,5)-\mu(p^1)}{l(4)-l(5)+1}\right\}\\ &= \min\left\{\frac{1}{4}, \frac{7}{20}, \frac{1}{12}, \frac{5}{12}\right\} = \frac{1}{12}\end{aligned}$$

і $p^2 = p^1 - \epsilon^2 l = (0, -1/3, -1/3, -2/3, 0, -1)$.

2. $\mu(p^2) = 4/3$, а граф $G(p^2)$ прадстаўлены на мал. 1.19. Граф $G(p^2)$ мае цыкл $Y = (3, 4, 6, 3)$, які і з'яўляецца цыклам мінімальнага сярэдняга кошту ў графе G. □

У заключэнне параграфа, як мы і абяцалі ў яго пачатку, параўнаем двойны алгарытм з алгарытмам Карпа. Ацэнка працаёмкасці апошняга на парадак лепшая, але памяці ён патрабуе таксама на парадак болей, што пры рашэнні задач вялікага памеру можа стаць вельмі сур'ёзнай праблемай. Акрамя таго, $O(n^2)$ — гэта ацэнка колькасці ітэрацый двойнага алгарытма ў разліку на найхужэйшы выпадак. На большасці прыкладаў алгарытм выконвае значна менш ітэрацый і таму яго складанасць таксама значна меншая чым $O(n^2 m)$. Няцяжка таксама зразумець, што складанасць метада Карпа $\Omega(nm)$, г. зн. што яна амаль адналькавая для ўсіх графаў з n вяршынямі і m дугамі. Таму можна чакаць (і гэта сапраўды так), што на практыцы двойны алгарытм у сярэднім працуе лепей чым алгарытм Карпа. Трэба аднак таксама заўважыць, што пры цэлалікавых коштах алгарытм

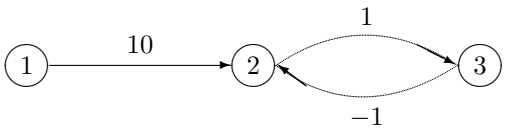


Малюнак 1.19: Граф $G(p^2)$

Карпа з'яўляецца "поўнасцю"цэлалікавым (усе вылічэнні праводзяцца над цэлымі лікамі), а двойны метад працуе з дробнымі лікамі (гл. практыкаванне 3).

1.9 Практыкаванні

1. Знайдзіце ўсе рашэнні ўраўненняў Бэлмана (1.10) для графа, які прадстаўлены на наступным малюнку.



2. Дакажыце, што калі ўсе вяршыні графа $G = (V, E)$ дасягаюцца з вяршыні $s \in V$ і граф G не мае цыклаў недадатнага кошту адносна функцыі $c : E \to \mathbb{R}$, то ўраўненні Бэлмана маюць адзінае рашэнне.

3. З якой пагрэшнасцю трэба праводзіць вылічэнні цэн вяршынь, каб знайсці дакладнае рашэнне задачы аб мінімальным сярэднім цыкле. Ацаніце памер лікаў $p(v)$ для $v \in V$.

4. Вырашыце прыклад 1.5 двойным метадам, а прыклад 1.6 метадам Карпа.

5. Дадзены арграф $G = (V, E)$, функцыя коштаў яго дуг $c : E \to \mathbb{R}$, вяршыня $s \in V$, з якой дасягаюцца ўсе астатнія вяршыні графа G.

Дакажыце, што наступная задача ЛП эквівалентна задачы пошука дрэва карацейшых шляхоў у графе G з коранем s:

$$\begin{aligned}
\sum_{(v,w)\in E} c(v,w)f(v,w) &\rightarrow \min \\
\sum_{(v,w)\in E(v,V)} f(v,w) &= \begin{cases} n-1, & v = s, \\ 1, & v \in V \setminus s\,, \end{cases} \\
f(v,w) &\geq 0, \quad \text{для ўсіх } (v,w) \in E\,.
\end{aligned}$$

Глава 2

Патокі ў сетках

Аўтамабільныя дарогі, чыгунка, электрычныя сеткі, сеткі камунікацый і многія іншыя фізічныя сеткі ўвайшлі ў наша паўсядзённае жыццё. Як следства, нават для неспецыяліста відавочна практычная важнасць і шырокая распаўсюджанасць сетак. Больш таго, так як фізічныя характарыстыкі сетак (напрыклад, ўмова захавання патока ў вузлах) дапускаюць вельмі простае матэматычнае апісанне, то практыкі могуць лёгка зразумець не толькі матэматычныя пастаноўкі патокавых аптымізацыйных задач, але таксама і тыя ідэі, якія ляжаць у аснове алгарытмаў іх рашэння.

2.1 Патокі і граф астатніх прапускных здольнасцей

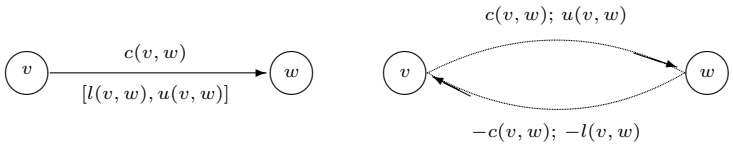
Сеткай называецца арыентаваны граф $G = (V, E)$ з функцыяй прапускных здольнасцей $u : E \to \mathbb{R} \cup \{\infty\}$. Без страты агульнасці дапускаем, што

- G не мае паралельных дуг, г. зн. $E \subseteq V \times V$;
- граф G з'яўляецца сіметрычным: з $(v, w) \in E$ вынікае, што $(w, v) \in E$.

Функцыя $f : E \to \mathbb{R}$ называецца *псеўдапатокам*, калі выконваюцца наступныя ўмовы

$$f(v, w) \leq u(v, w) \quad \text{для ўсіх } (v, w) \in E, \tag{2.1}$$
$$f(v, w) = -f(w, v) \quad \text{для ўсіх } (v, w) \in E. \tag{2.2}$$

Умова (2.1) задае *абмежаванні на прапускныя здольнасці дуг*: па дузе не можа працякаць паток большай велічыні чым яе прапускная здольнасць. *Умова антысіметрыі патока* (2.2) адлюстроўвае той факт, што паток велічыні x з v у w ёсць паток велічыні $(-x)$ з w у v. Адмоўныя значенні патока ўведзены толькі для зручнасці абазначэнняў. Заўважым, што ў гэтым выпадку няма патрэбы рабіць адрозненне паміж ніжнімі і верхнімі прапускнымі здольнасцямі дуг: прапускная здольнасць дугі (v, w) прадстаўляе



Малюнак 2.1:

ніжнюю прапускную здольнасць адваротнай дугі (w, v). Дольш дакладна, калі паток $f(v, w)$ па дузе (v, w) павінен задавальняць умове

$$l(v, w) \leq f(v, w) \leq u(v, w),$$

то мы замяняем дугу (v, w) парай супрацьлеглых дуг (v, w), (w, v) і азначаем іх кошты і прапускныя здольнасці так, як гэта прадстаўлена на малюнку 2.1.

З умоў (2.1) і (2.2) вынікае, што для псеўдапатока заўсёды выконваецца $u(v, w) + u(w, v) \geq 0$ для ўсіх дуг $(v, w) \in E$.

Для дадзенага псеўдапатока f азначым *лішак вяршыні* $v \in V$

$$e_f(v) = \sum_{(w,v)\in E} f(w, v)$$

як сумарны паток, які цячэ ў вяршыню v. Мы будзем гаварыць, што вяршыня v мае *лішак*, калі $e_f(v) > 0$, і мае *дэфіцыт*, калі $e_f(v) < 0$. Для вяршыні $v \in V$ *умова захавання патока* задаецца роўнасцю

$$e_f(v) = 0 \tag{2.3}$$

Цыркуляцыяй называецца псеўдапаток з нулявым лішкам у кожнай вяршыні.

Для псеўдапатока f *астатняя прапускная здольнасць* дугі $(v, w) \in E$ ёсць $u_f(v, w) = u(v, w) - f(v, w)$. *Граф астатніх прапускных здольнасцей* $G_f = (V, E_f)$ для псеўдапатока f мае мноства дуг $E_f = \{(v, w) \in E : u_f(v, w) > 0\}$.

Лема 2.1 *Няхай дадзены два псеўдапатокі f і g, прычым $e_f(v) > e_g(v)$. Тады існуе вяршыня w з $e_f(w) < e_g(w)$ і паслядоўнасць розных вяршынь $v = v_0, v_1, \ldots, v_{l-1}, v_l = w$, такая, што $(v_{i-1}, v_i) \in E_f$ і $(v_i, v_{i-1}) \in E_g$ для $1 \leq i \leq l$.*

Доказ. Азначым арграфы $G_+ = (V, E_+)$ і $G_- = (V, E_-)$, дзе

$$\begin{aligned} E_+ &= \{(x, y) \in E : g(x, y) > f(x, y)\}, \\ E_- &= \{(x, y) \in E : f(x, y) > g(x, y)\}. \end{aligned}$$

Тады $E_+ \subseteq E_f$, так як для $(x,y) \in E_+$ маем $f(x,y) < g(x,y) \leq u(x,y)$. Аналагічна, $E_- \subseteq E_g$. Больш таго, па антысіметрыі патока $(x,y) \in E_+$, калі і толькі калі $(y,x) \in E_-$. Таму дастаткова паказаць, што ў G_+ існуе просты шлях $v = v_0, v_1, \ldots, v_l = w$ з $e_f(w) < e_g(w)$.

Няхай S ёсць мноства вяршынь, якія дасягаюцца з v у графе G_+ і няхай $\overline{S} = V \setminus S$ (мноства $\overline{S}$ можа быць пустым). Для дугі $(x,y) \in E(S,\overline{S})$ справядліва няроўнасць $f(x,y) \geq g(x,y)$, бо інакш $y \in S$. Адсюль мы маем

$$\begin{aligned}
-\sum_{x\in S} e_g(x) &= \sum_{(x,y)\in E(S,V)} g(x,y) \\
&= \sum_{(x,y)\in E(S,\overline{S})} g(x,y) + \sum_{(x,y)\in E(S,S)} g(x,y) \\
&= \sum_{(x,y)\in E(S,\overline{S})} g(x,y) \quad \text{(па антысіметрыі)} \\
&\leq \sum_{(x,y)\in E(S,\overline{S})} f(x,y) \quad \text{(выконваецца для кожнага члена)} \\
&= \sum_{(x,y)\in E(S,\overline{S})} f(x,y) + \sum_{(x,y)\in E(S,S)} f(x,y) \quad \text{(па антысіметрыі)} \\
&= \sum_{(x,y)\in E(S,V)} f(x,y) \\
&= -\sum_{x\in S} e_f(x).
\end{aligned}$$

Так як $v \in S$ і $e_f(v) > e_g(v)$, то для нейкай вяршыні $w \in S$ павінна выконвацца $e_f(w) < e_g(w)$. □

Карыснай уласцівасцю псеўдапатокаў з'яўляецца той факт, што яны могуць быть раскладзены на невялікую колькасць *прымітыўных элементаў*. Такімі прымітыўнымі элементамі з'яўляюцца элементарныя патокі і цыркуляцыі.

Псеўдапаток g у сетцы G называецца *элементарным патокам (цыркуляцыяй)* велічыні ϵ, калі падграф G з мноствам дуг, на якіх g дадатны, з'яўляецца простым шляхам (цыклам) і на ўсіх дугах гэтага шляху (цыкла) велічыня патока роўна ϵ.

Тэарэма 2.1 *Для любога псеўдапатока f у сетцы G існуе такая сям'я элементарных патокаў і цыркуляцый $f_1, \ldots, f_k$, $k \leq m$, што*

$$f(v,w) = \sum_{i=1}^{k} f_i(v,w) \quad \textit{для ўсіх } (v,w) \in E. \tag{2.4}$$

Доказ. Індукцыяй па колькасці дуг з ненулявым патокам. Тэарэма выконваецца, калі $f \equiv 0$. Няхай f — ненулявы псеўдапаток. Дапусцім, што тэарэма выконваецца для ўсіх псеўдапатокаў f', такіх, што $|E(f')| < |E(f)|$. Тут $E(f) = \{(v,w) \in E : f(v,w) > 0\}$. Магчымы два выпадкі:

1) існуе вяршыня $v \in V$ з дадатным лішкам $e_f(v) > 0$;

2) $e_f(v) = 0$ для ўсіх $v \in V$ (f — цыркуляцыя).

Выпадак 1. Па леме 2.1, прыменена для $g \equiv 0$ і f, існуе вяршыня w з $e_f(w) < 0$ і просты шлях $w = v_0, v_1, \ldots, v_{l-1}, v_l = v$ у графе G_g, для якога $f(v_{i-1}, v_i) > 0$ для $i = 1, \ldots, l$. Няхай f_1 — элементарны паток, у якім па дугах гэтага шляху цячэ паток велічыні

$$\epsilon = f(v_{j-1}, v_j) = \min_{1 \le i \le l} f(v_{i-1}, v_i).$$

Азначым псеўдапаток f' па правілу:

$$f'(x, y) = f(x, y) - f_1(x, y) \quad \text{для ўсіх } (x, y) \in E.$$

Адзначым, што $f'(v_{j-1}, v_j) = 0$. Таму $|E(f')| < |E(f)|$ і па індукцыйнаму дапушчэнню псеўдапаток f' можа быць прадстаўлены як сума элементарных патокаў:

$$f'(x, y) = \sum_{i=2}^{k} f_i(x, y) \quad \text{для ўсіх } (x, y) \in E,$$

дзе $k \le m$. Таму

$$f(x, y) = \sum_{i=1}^{k} f_i(x, y) \quad \text{для ўсіх } (x, y) \in E.$$

Выпадак 2. Так як $f \not\equiv 0$, то існуе дуга $(v, w) \in E$ з ненулявым патокам $f(v, w)$. Для канкрэтнасці будзем лічыць, што $f(v, w) = -f(w, v) > 0$ (інакш трэба разглядаць дугу (w, v)). Пабудуем псеўдапаток $\overline{f}$ па правілу:

$$\overline{f}(v, w) = 0; \quad \overline{f}(x, y) = f(x, y) \quad \text{для ўсіх } (x, y) \in E \setminus (v, w).$$

Зразумела, што $e_{\overline{f}}(v) = -e_{\overline{f}}(w) = f(v, w) > 0$. Таму па леме 2.1, прыменена для $g \equiv 0$ і $\overline{f}$, існуе вяршыня w з $e_{\overline{f}}(w) < 0$ і просты шлях $w = v_0, v_1, \ldots, v_{l-1}, v_l = v$ у графе G_g, для якога $\overline{f}(v_{i-1}, v_i) > 0$ для ўсіх $1 \le i \le l$. Няхай f_1 — элементарная цыркуляцыя, у якой па дугах цыкла $w = v_0, v_1, \ldots, v_{l-1}, v_l, v_{l+1} = w$ цячэ паток велічыні

$$\epsilon = \min_{1 \le i \le l+1} f(v_{i-1}, v_i) > 0.$$

Далей доказ такі ж, як і ў выпадку 1. □

2.2 Фармулёўкі класічных патокавых задач

2.2.1 Задача аб максімальным патоку

Дадзены сетка (G, u), у якой вылучаны дзве вяршыні $s, t \in V$; s называецца *вытокам*, а t — *сцёкам Пераднатокам* называецца псеўдапаток f з неадмоўным лішкам для ўсіх вяршынь, за выключэннем s. *Патокам* f у сетцы G называецца псеўдапаток, які задавальняе ўмове захавання патока (2.3) ва ўсіх вяршынях акрамя s і t. *Велічынёй* патока f называецца колькасць патока, які ўцякае ў сцёк $|f| = e_f(t)$. *Максімальны паток* — гэта паток максімальнай велічыні. *Задача аб максімальным патоку* ёсць задача пошука патока максімальнай велічыні ў дадзенай *патокавай сетцы* (G, u, s, t).

2.2.2 Задача аб цыркуляцыі мінімальнага кошту

Функцыя кошту ёсць сапраўдная функцыя на мностве дуг $c : E \to \mathbb{R}$. Без страты агульнасці мы дапускаем, што яна антысіметрычная:

$$c(v, w) = -c(w, v) \quad \text{для ўсіх } (v, w) \in E. \tag{2.5}$$

Кошт псеўдапатока f азначаецца па формуле:

$$c(f) \stackrel{\text{def}}{=} \sum_{(v,w)\in E: f(v,w)\geq 0} f(v, w)c(v, w).$$

Задача аб цыркуляцыі мінімальнага кошту заключаецца ў пошуку ў сетцы (G, u, c) цыркуляцыі, якая мае мінімальны кошт.

Заўважым, што задача аб максімальным патоку з'яўляецца прыватным выпадкам задачы аб цыркуляцыі мінімальнага кошту. Каб убачыць гэта, разгледзім патокавую сетку (G, u, s, t), да якой дададзім пару новых дуг (s, t) і (t, s) с $u(s, t) = 0$, $u(t, s) = \infty$. Кошты зыходных дуг вызначым роўнымі нулю, а $c(s, t) = -c(t, s) = 1$. Цыркуляцыя мінімальнага кошту ў атрыманай сетцы (G^{st}, u, c), абмежаваная на дугі зыходнай сеткі, з'яўляецца максімальным патокам у (G, u, s, t).

2.2.3 Транспартная задача

У дапаўненне да фукцый кошту c і прапускных здольнасцей дуг u дадзена *функцыя попыту* $d : V \to \mathbb{R}$ такая, што $\sum_{v\in V} d(v) = 0$. Псеўдапаток называем *дапушчальным*, калі выконваюцца наступная *ўмова захавання*:

$$e_f(v) = d(v) \quad \text{для ўсіх } v \in V. \tag{2.6}$$

Транспартная задача ёсць задача, у якой у *транспартнай сетцы* (G, u, c, d) трэба знайсці дапушчальны псеўдапаток мінімальнага кошту.

Відавочна, што задача аб цыркуляцыі мінімальнага кошту з'яўляецца спецыяльным выпадкам транспартнай задачы, у якой $d(v) = 0$ для ўсіх

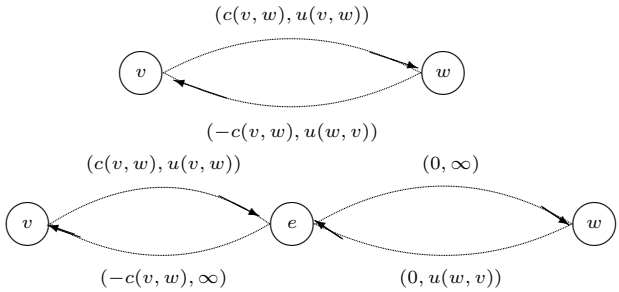
$v \in V$. У сваю чаргу, транспартную задачу ў сетцы (G, u, c, d) можна пераўтварыць у задачу аб цыркуляцыі мінімальнага кошту. Для гэтага да G дададзім новую вяршыню $s \notin V$ і мноства дуг: $\{(s, v), (v, s) : v \in V,\ d(v) \neq 0\}$. Функцыя кошту і прапускныя здольнасці даазначаюцца наступным чынам:

- калі $d(v) > 0$, то
$$u(v, s) = d(v),\ c(v, s) = -\infty, \quad u(s, v) = 0,\ c(s, v) = \infty;$$
- калі $d(v) < 0$, то
$$u(s, v) = -d(v),\ c(s, v) = -\infty, \quad u(v, s) = 0,\ c(v, s) = \infty.$$

Няцяжка пераканацца, што абмежаванне цыркуляцыі ў пабудаванай сетцы на дугі зыходнай сеткі будзе дапушчальным рашэннем транспартнай задачы, прычым аптымальнай цыркуляцыі адпавядае аптымальнае дапушчальнае рашэнне транспартнай задачы.

Далей мы таксама будзем разглядаць варыянт транспартнай задачы, калі ўсе прапускныя здольнасці дуг роўны нулю або бясконцасці. Такі варыянт задачы называем *транспартнай задачай без абмежаванняў на прапускныя здольнасці*.

Звядзем задачу аб цыркуляцыі минимальнага кошту на сетцы (G, u, c) да транспартнай задачы без абмежаванняў на прапускныя здольнасці дуг. Спачатку для кожнай пары дуг $e = (v, w)$ і $\overline{e} = (w, v)$, для якой абодва лікі $u(v, w)$ і $u(w, v)$ концыя, дададзім новую вяршыню e и чатыры дугі (v, e), (e, v), (e, w), (w, e) (гл. мал. 2.2.3). Атрыманую сетку абазначым праз $\overline{G}, \overline{u}, \overline{c}$.



Малюнак 2.2: Трансфармацыя дуг задачы аб цыркуляцыі

Потым пакладзем $d(v) = 0$ для ўсіх $v \in V$ і для кожнай дугі (v, w), для якой $u(v, w)$ ёсць концы ненулявы лік, выканаем наступныя аперацыі:

$$\begin{aligned} d(v) &:= d(v) - u(v, w), \\ d(w) &:= d(w) - u(w, v), \\ u(v, w) &:= 0. \end{aligned}$$

2.3 Крытэрыі аптымальнасці

Крытэрыі аптымальнасці з'яўляюца той асновай, на якой будуюцца алгарытмы рашэння аптымізацыйных задач.

2.3.1 Транспартная задача

Пачнем з простай лемы.

Лема 2.2 *Няхай f і g — дапушчальныя псеўдапатокі ў транспартнай сетцы (G, u, c, d). Тады $g-f$ ёсць дапушчальная цыркуляцыя ў сетцы (G, u_f, c).*

Доказ. Так як $g(v,w) - f(v,w) \le u(v,w) - f(v,w) = u_f(v,w)$, то абмежаванні на прапускныя здольнасці выконваюцца. Так як

$$e_{g-f}(v) = e_g(v) - e_f(v) = d(v) - d(v) = 0,$$

то таксама выконваецца ўмова захавання патока ва ўсіх вяршынях $v \in V$. □

Зараз мы сфармулюем два крытэрыі аптымальнасці псеўдапатока.

Тэарэма 2.2 *Дапушчальны псеўдапаток f у транспартнай сетцы (G, u, c, d) аптымальны тады і толькі тады, калі яго граф астатніх прапускных здольнасцей G_f не мае цыклаў адмоўнага кошту.*

Доказ. **Неабходнасць**. Няхай ў графе G_f ёсць цыкл

$$\Gamma = (v_0, v_1, \ldots, v_{l-1}, v_l = v_0)$$

адмоўнага кошту $c(\Gamma) < 0$. Для

$$\epsilon \stackrel{\text{def}}{=} \min_{1 \le i \le l} u_f(v_{i-1}, v_i) > 0$$

пабудуем дапушчальны псеўдапаток f' наступным чынам:

$$\begin{aligned} f'(v_{i-1}, v_i) &= f(v_{i-1}, v_i) + \epsilon, \quad i = 1, \ldots, l, \\ f'(v_i, v_{i-1}) &= -f'(v_{i-1}, v_i), \quad i = 1, \ldots, l, \\ f'(v,w) &= f(v,w), \quad (v,w) \in E \setminus E(\Gamma). \end{aligned}$$

Так як $c(f') = c(f) + \epsilon c(\Gamma) < c(f)$, то f не з'яўляецца аптымальным.

Дастатковасць. Дапусцім, што граф G_f не мае цыклаў адмоўнага кошту. Няхай h — аптымальны дапушчальны псеўдапаток ў сетцы (G, u, c, d). Па леме 2.2 псеўдапаток $(h-f)$ з'яўляецца цыркуляцыяй у сетцы (G, u_f, c), а па тэарэме 2.1 існуе такая сям'я элементарных цыркуляцый $g_1, \ldots, g_k$ у (G, u_f, c), што

$$h(v,w) - f(v,w) = \sum_{i=1}^{k} g_i(v,w) \quad \text{для ўсіх } (v,w) \in E,$$

Так як

$$\begin{aligned} 0 &\geq c(h) - c(f) \\ &= \frac{1}{2}\sum_{(v,w)\in E} c(v,w)h(v,w) - \frac{1}{2}\sum_{(v,w)\in E} c(v,w)f(v,w) \\ &= \frac{1}{2}\sum_{(v,w)\in E_f} c(v,w)(h(v,w) - f(v,w)) \\ &= \frac{1}{2}\sum_{(v,w)\in E_f} c(v,w)\sum_{i=1}^{k} g_i(v,w) \\ &= \sum_{i=1}^{k}\sum_{(v,w)\in E} \frac{1}{2}c(v,w)g_i(v,w) \\ &= \sum_{i=1}^{k} c(g_i) \geq 0. \end{aligned}$$

Адсюль маем, што f — таксама аптымальны дапусцімы псеўдапаток. □

Непасрэдна з лемы 1.2.

Лема 2.3 *Дапушчальны псеўдапаток f з'яўляецца аптымальным у транспартнай сетцы (G, u, c, d) тады і толькі тады, калі ён аптымальны ў сетцы (G, u, c_p, d) для кожнай функцыі цэн p.*

Другі крытэрый аптымальнасці дапушчальнага псеўдапатока наступны.

Тэарэма 2.3 *Дапушчальны псеўдапаток f з'яўляецца аптымальным у транспартнай сетцы (G, u, c, d) тады і толькі тады, калі існуе такая функцыя цэн p, што выконваецца ўмова дапаўняючай няжорсткасці*

$$c_p(v,w) \geq 0 \quad \textit{для ўсіх } (v,w) \in E_f. \tag{2.7}$$

Доказ. Па тэарэме 2.2 псеўдапаток f аптымальны тады і толькі тады, калі граф G_f не мае цыклаў адмоўнага кошту. Згодна выніку 1.1 граф G_f не мае цыклаў адмоўнага кошту тады і толькі тады, калі існуе такая функцыя $p : V \to \mathbb{R}$, якая задавальняе ўмове (2.7). □

Транспартная задача можа не мець аптымальнага рашэння.

Тэарэма 2.4 *Транспартная задача на сетцы (G, u, c, d) мае аптымальнае рашэнне тады і толькі тады, калі яна мае дапушчальнае рашэнне і граф G не мае цыкла адмоўнага кошту, усе дугі якога маюць бясконцую прапускную здольнасць.*

2.3.2 Задача аб максімальным патоку

Выкарыстоўваючы тэарэму 2.2, атрымаем два крытэрыі аптымальнасці.

Тэарэма 2.5 *Паток f з'яўляецца максімальным патокам у сетцы (G, u, s, t) тады і толькі тады, калі ў графе G_f адсутнічаюць шляхі з s у t.*

Доказ. Даазначым паток f да цыркуляцыі ў сетцы (G^{st}, u, c), паклаўшы $f(t, s) = e_f(t), f(s, t) = -f(t, s)$. Па тэарэме 2.2 цыркуляцыя f аптымальна тады і толькі тады, калі ў графе G_f^{st} адсутнічаюць цыклы адмоўнага кошту. Па пабудове сеткі (G^{st}, u, c) граф G_f^{st} мае цыкл адмоўнага кошту тады і толькі тады, калі ён ўтрымлівае дугу (t, s), г.зн. калі ў G_f ёсць шлях з s у t. □

Няхай $S \subseteq V$. Нагадаем, што непустое мноства $E(S, V \setminus S)$ называецца *разрэзам*. Разрэз $E(S, V \setminus S)$ называецца s, t-*разрэзам*, калі S з'яўляецца $s, \bar{t}$-*мноствам*, г. зн. $s \in S$, $t \notin S$. *Велічынёй разрэза* называецца сума прапускных здольнасцей яго дуг, г. зн. лік $\delta_u(S) = \sum_{(v,w)\in E(S,V\setminus S)} u(v, w)$. Фундаментальнай тэарэмай аб патоках у сетках з'яўляецца наступная тэарэма.

Тэарэма 2.6 (Форд-Фалкерсан) *Велічыня максімальнага патока ў патокавай сетцы (G, u, s, t) роўна велічыні мінімальнага s, t-разрэза.*

Доказ. Няхай f — адвольны паток у сетцы (G, u, s, t), а $E(S, V \setminus S)$ — адвольны s, t-разрэз. Склаўшы роўнасці

$$\begin{aligned} |f| &= -e_f(s), \\ 0 &= -e_f(v), \quad v \in S \setminus s, \end{aligned}$$

атрымаем

$$\begin{aligned} |f| &= -\sum_{v\in S} e_f(v) \\ &= -\sum_{v\in S} \sum_{(w,v)\in E} f(w, v) \\ &= -\sum_{(w,v)\in E(V,S)} f(w, v) \\ &= \sum_{(v,w)\in E(S,V)} f(v, w) \qquad (2.8) \\ &= \sum_{(v,w)\in E(S,V\setminus S)} f(v, w) + \sum_{(v,w)\in E(S,S)} f(v, w) \\ &= \sum_{(v,w)\in E(S,V\setminus S)} f(v, w) \\ &\le \sum_{(v,w)\in E(S,V\setminus S)} u(v, w) \\ &= \delta_u(S). \end{aligned}$$

Няхай зараз f — максімальны паток, а S ёсць мноства вяршынь графа G_f, дасягаемых з вытока s. Відавочна, што $s \in S$, а па тэарэме 2.5 $t \notin S$. Акрамя таго, $u(v,w) = f(v,w)$ для ўсіх такіх дуг $(v,w) \in E(S, V \setminus S)$. Але ў гэтым выпадку няроўнасць ў (2.8) пераўтвараецца ў роўнасць і тады $|f| = \delta_u(S)$. □

У якасці выніку тэарэмы Форда-Фалкерсана сфармулюем наступны крытэрый існавання цыркуляцыі у сетцы (G,u).

Тэарэма 2.7 (Гофман) *У сетцы (G,u) існуе цыркуляцыя тады і толькі тады, калі*

$$\delta_u(X) \geq 0 \quad \textit{для ўсіх } X \subseteq V. \tag{2.9}$$

Доказ. **Неабходнасць**. Няхай у сетцы (G,u) існуе цыркуляцыя f. Тады

$$\begin{aligned} \delta_u(X) &= \sum_{(v,w)\in E(X,V\setminus X)} u(v,w) \\ &\geq \sum_{(v,w)\in E(X,V\setminus X)} f(v,w) \\ &= \sum_{(v,w)\in E(X,V)} f(v,w) \\ &= -\sum_{v\in X} e_f(v) = 0. \end{aligned}$$

Дастатковасць. Для прастаты мы абмяркуем выпадак канцазначнай функцыі $u : E \to \mathbb{R}$ (агульны выпадак застаецца чытачу). Пабудуем псеўдапаток f па правілу:

$$\text{калі } u(v,w) \geq -u(w,v), \text{ то } f(v,w) = -u(w,v) \text{ і } f(w,v) = u(w,v).$$

Калі $e_f(v) = 0$ для ўсіх $v \in V$, то f — цыркуляцыя. Інакш мноствы $S = \{v \in V : e_f(v) > 0\}$ і $T = \{v \in V : e_f(v) < 0\}$ не пустыя. Пабудуем арграф $G' = (V', E')$, дзе

$$V' \stackrel{\text{def}}{=} V \cup \{s,t\}, \quad E' \stackrel{\text{def}}{=} E \cup \{(s,v) \,:\, v \in S\} \cup \{(v,t) \,:\, v \in T\}.$$

Азначым прапускныя здольнасці дуг наступным чынам:

$$\begin{aligned} u'(s,v) &= e_f(v), \quad v \in S, \\ u'(v,t) &= -e_f(v), \quad v \in T, \\ u'(v,w) &= u(v,w) - f(v,w), \quad (v,w) \in E. \end{aligned}$$

Няхай $M = \sum_{v\in S} e_f(v)$. Няцяжка бачыць, што ў патокавай сетцы (G', u', s, t) існуе паток x велічыні M тады і толькі тады, калі функцыя $f + x$ (абмежаваная на E) з'яўляецца цыркуляцыяй ў (G,u). Па тэарэме 2.6 велічыня максімальнага патока ў (G', u', s, t) роўна M тады і толькі тады, калі

$$\delta_{u'}(X \cup \{s\}) \geq M \quad \text{для ўсіх } X \subseteq V. \tag{2.10}$$

Але

$$\begin{aligned}
\delta_{u'}(X\cup\{s\}) &= \sum_{(v,w)\in E'(X\cup\{s\},V\cup\{t\}\setminus X)} u'(v,w)\\
&= \sum_{(s,w):w\in S\setminus X} u'(s,w)+\sum_{(v,t):v\in X\cap T} u'(v,t)+\\
&\quad \sum_{(v,w)\in E(X,V\setminus X)} u'(v,w)\\
&= \sum_{w\in S\setminus X} e_f(w)-\sum_{v\in X\cap T} e_f(v)+\\
&\quad \sum_{(v,w)\in E(X,V\setminus X)} (u(v,w)-f(v,w))\\
&= \sum_{v\in S\setminus X} e_f(v)-\sum_{v\in X\cap T} e_f(v)+\\
&\quad \delta_u(X)-\sum_{(v,w)\in E(X,V)} f(v,w)\\
&= \sum_{v\in S\setminus X} e_f(v)-\sum_{v\in X\cap T} e_f(v)+\delta_u(X)+\sum_{v\in X} e_f(v)\\
&= \sum_{v\in S\setminus X} e_f(v)-\sum_{v\in X\cap T} e_f(v)+\delta_u(X)+\\
&\quad \sum_{v\in X\cap S} e_f(v)+\sum_{v\in X\cap T} e_f(v)\\
&= \sum_{v\in S} e_f(v)+\delta_u(X)\\
&= M+\delta_u(X).
\end{aligned}$$

Адсюль маем, што ўмовы (2.9) і (2.10) эквівалентны. □

На заканчэнне сфармулюем два камбінаторныя следствы з тэарэмы Форда-Фалкерсана.

Тэарэма 2.8 (Менгер) *У арграфе $G=(V,E)$ існуе k шляхоў з s у t, якія не маюць агульных дуг, тады і толькі тады, калі $\rho(X)\geq k$ для усіх $s,\bar{t}$-падмностваў $X\subseteq V$.*

Доказ. Разгледзець патокавую сетку (G,u,s,t), дзе прапускныя здольнасці ўсіх дуг роўны 1, і прымяніць тэарэму Форда-Фалкерсона. □

Тэарэма 2.9 (Кёніг) *Максімальная моц паразлучэння ў двухдольным графе $G=(V\cup W,E)$ роўна мінімальнай колькасці вяршынь, якія пакрываюць усе рэбры.*

Доказ. З'арыентуем рэбры графа G у напрамку ад V да W. Затым дададзім да G дзве новыя вяршыні $s, t \notin V \cup W$ і два мноствы дуг

$$\{(s, v) : v \in V\}, \quad \{(v, t) : v \in W\}.$$

Прапускныя здольнасці u ўсіх новых дуг роўны 1, а дуг з E — ∞. Атрыманую патокавую сетку абазначым праз $(G^{s,t}, u, s, t)$. Паміж паразлучэннямі ў графе G і цэлалікавымі патокамі ў сетцы $(G^{s,t}, u, s, t)$ існуе ўзаемна адназначная адпаведнасць: рабро $(v, w) \in E$ належыць паразлучэнню тады і толькі тады, калі $f(s, v) = f(v, w) = f(w, t) = 1$. Зразумела, што максімальнаму патоку велічыні k адпавядае паразлучэнне моцы k. Па тэарэме 2.6 існуе $s, \bar{t}$-мноства X, для якога $\delta_u(X) = k$. Адзначым, што ніводная дуга з бясконцай прапускной здольнасцю не належыць $E(X, V \cup W \setminus X)$. Таму мноства $(V \setminus X) \cup (W \cap X)$ мае моц k і пакрывае ўсе рэбры. □

2.3.3 Прыблізная аптымальнасць

Разгледзім транспартную задачу на сетцы (G, u, c, d). Ключавое паняцце прыблізнай аптымальнасці атрымліваецца ў выніку паслаблення ўмовы дапаўняючай няжорсткасці ў тэарэме 2.3. Для дадзенага $\epsilon \geq 0$ псеўдапаток f называецца *ϵ-аптымальным адносна функцыі цэн p*, калі

$$c_p(v, w) \geq -\epsilon \quad \text{для ўсіх } (v, w) \in E_f. \tag{2.11}$$

Псеўдапаток f называецца *ϵ-аптымальным*, калі f з'яўляецца ϵ-аптымальным адносна нейкай функцыі цэн p.

Важная ўласцівасць ϵ-аптымальнасці ў тым, што калі кошты дуг цэлалікавыя і ϵ дастаткова малы лік, то ϵ-аптымальны дапушчальны псеўдапаток з'яўляецца аптымальным.

Тэарэма 2.10 *Калі кошты ўсіх дуг цэлалікавыя і $\epsilon < 1/n$, то дапушчальны ϵ-аптымальны псеўдапаток з'яўляецца аптымальным.*

Доказ. Разгледзім просты цыкл у графе G_f. З ϵ-аптымальнасці псеўдапатока f вынікае, што яго прыведзены кошт не меншы чым $-\frac{\epsilon}{n} > -1$. Але прыведзены кошт цыкла роўны яго зыходнаму кошту, які павінен быць цэлалікавым і, такім чынам, неадмоўным. З тэарэмы 2.2 выводзім, што f — аптымальны псеўдапаток. □

Азначэнне ϵ-аптымальнасці матывуе пастаноўку наступных дзвюх задач:

(31) Для дадзенага псеўдапатока f і канстанты $\epsilon \geq 0$ знайсці такую функцыю цэн p, што f з'яўляецца ϵ-аптымальным адносна p, або паказаць, што такой функцыі цэн не існуе (г. зн., што f не з'яўляецца ϵ-аптымальным).

(32) Для дадзенага псеўдапатока f знайсці найменшае $\epsilon \geq 0$, такое, што f з'яўляецца ϵ-аптымальным. Такое ϵ абазначым праз $\epsilon(f)$.

Разгледзім задачу (31). Вызначым новыя кошты $c^\epsilon(v,w) \stackrel{\text{def}}{=} c(v,w) + \epsilon$ для ўсіх $(v,w) \in E$.

Тэарэма 2.11 *Псеўдапаток f з'яўляецца ϵ-аптымальным, тады і толькі тады, калі граф G_f не мае цыклаў адмоўнага кошту адносна функцыі кошту c^ϵ.*

Доказ. Калі граф G_f не мае цыклаў адмоўнага кошту адносна c^ϵ, існуе функцыя цэн p, што $c^\epsilon(v,w) \geq 0$ для ўсіх $(v,w) \in E$, ці $c(v,w) \geq -\epsilon$ для ўсіх $(v,w) \in E$.

Наадварот, калі f — ϵ-аптымальны псеўдапаток, то існуе функцыя цэн p, што $c_p(v,w) \geq -\epsilon$ для ўсіх $(v,w) \in E$, ці $c^\epsilon_p(v,w) \geq 0$ для ўсіх $(v,w) \in E$. Але ў гэтым выпадку (следства 1.1) граф G_f не мае цыклаў адмоўнага кошту адносна c^ϵ. □

З тэарэмы 2.11 вынікае, што задача (31) эквівалентна задачы пошука адмоўнага цыкла і можа быць вырашана алгарытмам Форда-Бэлмана за час $O(nm)$.

Разгледзім цяпер задачу (32).

Тэарэма 2.12 *Для любога псеўдапатока f мае месца формула*

$$\epsilon(f) = \max\{0, -\mu(G_f, c)\}.$$

Доказ. Непасрэдна з лемы 1.8. □

Наступная тэарэма паказвае, што калі прыведзены кошт нейкай дугі істотна большы за бягучую велічыню памылкі ϵ, то любы аптымальны дапушчальны псеўдапаток мае тую ж велічыню патока на гэтай дузе, што і бягучы псеўдапаток.

Тэарэма 2.13 *Няхай дадзены канстанты $\epsilon > 0$ і $\epsilon' \geq 0$. Калі дапушчальны псеўдапаток f з'яўляецца ϵ-аптымальным адносна функцыі цэн p, і для дугі $(v,w) \in E$ выконваецца ўмова $|c_p(v,w)| \geq n(\epsilon + \epsilon')$, то тады $f(v,w) = f'(v,w)$ для кожнага дапушчальнага ϵ'-аптымальнага псеўдапатока f'.*

Доказ. З прычыны антысіметрыі функцыі кошту дастаткова даказаць тэарэму для выпадку $c_p(v,w) \geq n(\epsilon + \epsilon')$. Няхай f' ёсць дапушчальны псеўдапаток, для якога $f'(v,w) \neq f(v,w)$. Так як $c_p(v,w) > \epsilon$, то $f(v,w) = -u(w,v)$, і таму з $f'(v,w) \neq f(v,w)$ маем $f'(v,w) > f(v,w)$. Дакажам, што f' не з'яўляецца ϵ'-аптымальным і, тым самым, тэарэма будзе даказана.

Разгледзім падграф $G_> = (V, E_>)$, дзе

$$E_> \stackrel{\text{def}}{=} \{(x,y) \in E \,:\, f'(x,y) > f(x,y)\},$$

графа G_f. Зазначым, што (v,w) з'яўляецца дугой графа $G_>$. Так як $(f'-f)$ ёсць дапушчальная цыркуляцыя ў сетцы (G,u_f,c), то $G_>$ павінен мець просты цыкл Γ, які праходзіць праз дугу (v,w). Няхай Γ мае l дуг. Так як усе дугі цыкла Γ належаць G_f , то яго кошт не меншы чым

$$c_p(v,w)-(l-1)\epsilon \geq n(\epsilon+\epsilon')-(n-1)\epsilon > n\epsilon'.$$

Зараз разгледзім цыкл $\overline{\Gamma}$, які атрымліваецца з Γ заменай усіх дуг на адваротныя. Заўважым, што $\overline{\Gamma}$ ёсць цыкл графа $G_{f'}$. Па антысіметрыі

$$c(\overline{\Gamma}) = -c(\Gamma) < -n\epsilon'.$$

Таму $\mu(G_{f'},c) < -\epsilon'$, а па тэарэме 2.12 вынікае, што $\epsilon(f') > \epsilon'$. □

Каб сфармуляваць важны вынік тэарэмы 2.12, увядзем наступнае азначенне. Дугу (v,w) называем *ϵ-фіксаванай*, калі паток па ёй аднолькавы для ўсіх дапушчальных ϵ-аптымальных псеўдапатокаў.

Вынік 2.1 *Няхай $\epsilon > 0$ і дапушчальны псеўдапаток f з'яўляецца ϵ-аптымальным адносна функцыі цэн p. Калі $|c_p(v,w)| \geq 2n\epsilon$, то дуга (v,w) з'яўляецца ϵ-фіксаванай.*

Доказ. Непасрэдна з тэарэмы 2.13, калі пакласці $\epsilon' = \epsilon$. □

Азначым мноства F_ϵ усіх ϵ-фіксаваных дуг. Наступная тэарэма з'яўляецца асноўным інструментам пры атрыманні палінаміяльных ацэнак складанасці алгарытмаў.

Тэарэма 2.14 *Няхай $\epsilon = \epsilon(f)$ для нейкага дапушчальнага псеўдапатока f і $\epsilon' \leq \epsilon/2n$. Тады F_ε з'яўляецца ўласным падмноствам $F_{\epsilon'}$.*

Доказ. Так як кожны ϵ'-аптымальны дапушчальны псеўдапаток з'яўляецца таксама ϵ-аптымальным, то $F_\epsilon \subseteq F_{\epsilon'}$. Каб паказаць, што ўлучэнне з'яўляецца ўласным, нам трэба паказаць, што ёсць ϵ'-фіксаваная дуга, якая не з'яўляецца ϵ-фіксаванай. Так як $\epsilon = \epsilon(f)$, то існуе такая функцыя цэн p, што f з'яўляецца ϵ-аптымальным адносна p. Акрамя таго, граф G_f мае просты цыкл Γ, усе дугі якога маюць прыведзены кошт $-\epsilon$. Так як павелічэнне псеўдапатока f уздоўж Γ захоўвае ϵ-аптымальнасць, то дугі Γ не з'яўляюцца ϵ-фіксаванымі.

Пакажам, што па меншай меры адна дуга цыкла Γ з'яўляецца ϵ'-фіксаванай. Няхай f' ёсць дапушчальны псеўдапаток, які ϵ'-аптымальны адносна функцыі цэн p'. Так як сярэдні кошт цыкла Γ роўны $-\epsilon$, то ён мае дугу (v,w), для якой $c_{p'}(v,w) \leq -\epsilon \leq -2n\epsilon'$. Па выніку 2.1 дуга (v,w) з'яўляецца ϵ'-фіксаванай. □

Глава 3

Задача аб максімальным патоку

Нагадаем, што задача аб максімальным патоку ёсць задача пошука патока максімальнай велічыні ў дадзенай патокавай сетцы (G, u, s, t) (гл. § 2.2.1). Трэба таксама адзначыць, што працэдура для вырашэння задачы аб максімальным патоку часта прымяняюцца як падпраграма пры рашэнні шэрагу больш складаных задач на графах и сетках. Звычайна такая падпраграмма выклікаецца шматразова, і таму яна павінна выконвацца вельмі хутка. Гэтым абумоўлена тое, што распрацоўцы эфектыўных алгарытмаў для вырашэння задачы аб максимальным патоку ўдзялася шмат увагі.

3.1 Алгарытм адметак

Для вырашэння задачы аб максімальным патоку з тэарэмы 2.5 непасрэдна атрымліваем алгарытм адметак Форда-Фалкерсона, які прадстаўлены на мал. 3.1.

Алгарытм адметак выкарыстоўвае дзве працэдуры: *find_path* і *augment*. Працэдура *find_path* у графе G_f знаходзіць шлях з вытоку s у сцёк t, які задаецца з дапамогай указальнікаў *parent*. Калі пасля заканчэння працэдуры *find_path* метка $parent(t) = \mathbf{nil}$, то ў графе G_f няма шляхоў з s у t. У гэтым выпадку бягучы паток з'яўляецца аптымальным і алгарытм адметак спыняецца. Працэдура *augment* знаходзіць мінімальную астатнюю прапускную здольнасць ϵ дугаў знойдзенага шляху і дадаткова праводзіць ϵ адзінак патока ўздоўж гэтага шляху.

Алгарытм адметак пакідае некаторую свабоду ў выбары шляху для павелічэння патока. Дзініц, а затым Эдмандс і Карп, прапанавалі мадыфікацыю алгарытма, у якой павелічэнне патока праводзіцца ўздоўж шляхоў з s у t карацейшай даўжыні. Гэтую мадыфікацыю мы называем алгарытмам Эдмандса-Карпа (алгарытм Дзініца выкарыстоўвае, акрамя гэтай, іншыя ідэі і з'яўляецца больш эфектыўным). Яе мы атрымаем, калі ў працеду-

```
labeling_algorithm(V, E, u, s, t, f) // f — пачатковы паток,
// напрыклад, f(v, w) = 0 для ўсіх (v, w) ∈ E
{
 for (;;) {
   find_path(G, u, s, t, f, parent);
   if (parent(t) = nil) return;
   augment(u, s, t, f, parent);
 }
}

find_path(G, u, s, t, f, parent)
{
 Q := {s};
 for (v ∈ V) parent(v) := nil; parent(s) := s;
 for (;Q ≠ ∅;) {
   v ← Q; // выбіраем вяршыню з Q
   for ((v, w) ∈ E_f(v, V)) if (parent(w) = nil) {
     Q ← w; parent(w) := v;
     if (w = t) return;
   }
 }
}

augment(u, s, t, f, parent)
{
 ϵ := ∞;
 for (v := parent(w := t); v ≠ s; v := parent(w := v))
      ϵ := min{ϵ, u_f(v, w)};
 // павялічваем паток
 for (v := parent(w := t); v ≠ s; v := parent(w := v))
      {f(v, w) := f(v, w) + ϵ; f(w, v) := −f(w, v);}
}
```

Малюнак 3.1: Алгарытм адметак

ры *find_path* спіс Q арганізаваць ў выглядзе чаргі. Як мы ўбачым пазней, складанасць алгарытма Эдмандса-Карпа палінаміяльна залежыць ад n і m.

Няхай $\sigma_f(v)$ ёсць адлегласць (даўжыня карацейшага шляху) ад s да v у графе G_f (калі няма шляху з s у v, то $\sigma_f(v) = \infty$. У графе G_f разгледзім шлях P карацейшай даўжыні з s у t. Ясна, што

$$\sigma_f(w) = \sigma_f(v) + 1 \text{ для кожнай дугі } (v, w) \text{ шляху } P \tag{3.1}$$

Лема 3.1 *Няхай f і g адпаведна патокі ў пачатку і канцы нейкай ітэрацыі алгарытма Эдмандса-Карпа. Тады $\sigma_f(v) \leq \sigma_g(v)$ для ўсіх $v \in V$.*

Доказ. Разгледзім, як зменіцца граф астатніх прапускных здольнасцей G_f пасля павелічэння патока f ўздоўж карацейшага шляху P ад s да t на велічыню $\epsilon = \min\{u_f(v,w) : (v,w) \in P\}$. Так як паток мяняецца толькі на дугах шляху P (і адваротных да іх), то і граф астатніх прапускных здольнасцей можа змяніцца таксама толькі на гэтых дугах. Менавіта, у графе G_g могуць ўзнікнуць новыя дугі, якія з'яўляюцца адваротнымі да дуг шляху P, і знікнуць *крытычныя дугі* (тыя, для якіх $u_f(v,w) = \epsilon$) шляху P. Адлегласць да v з s можа паменшыцца толькі тады, калі дабавіцца дуга (x,y), для якой $\sigma_f(y) > \sigma_f(x) + 1$, што немагчыма па (3.1). □

Лема 3.2 *На працягу ўсяго часу выканання алгарытма Эдмандса-Карпа дуга (v,w) можа быць крытычнай не больш чым n разоў.*

Доказ. Пры павелічэнні патока f па крытычнай дузе (v,w) яна знікае з G_f. Дапусцім, што пазней пры новым павелічэнні бягучага патока g дуга (v,w) з'яўляецца зноў. Тады $\sigma_g(v) = \sigma_g(w) + 1$. Па леме 3.1 мы маем

$$\sigma_f(v) + \sigma_f(w) = 2\sigma_f(w) - 1 \leq 2\sigma_g(w) - 1 = \sigma_g(v) + \sigma_g(w) - 2.$$

Так як адлегласць ад s да v не пераўзыходзіць $n-1$, то відавочна, што лема справядліва. □

Тэарэма 3.1 *Складанасць алгарытму Эдмандса-Карпа $O(nm^2)$.*

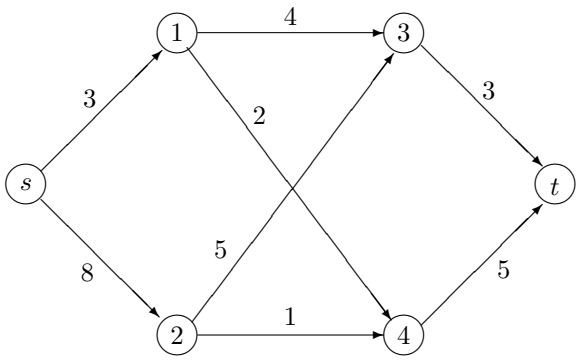
Доказ. Па леме 3.2 алгарытм робіць не больш чым nm павелічэнняў патока. Кожнае павелічэнне можа быць выканана за час $O(m)$ ($O(m)$ патрабуецца працэдуры *find_path* і $O(n)$ — працэдуры *augment*). Таму агульная складанасць алгарытма $O(nm^2)$. □

Прыклад 3.1 *Разгледзім патокавую сетку, якая прадстаўлена на мал. 3.2.*

Пачынаем рашаць задачу з нулявога патока. Ніжэй прыведзены ітэрацыі алгарытма адметак.

1. $Q = \{s,1,2,3,4,t\}$, $parent = (\mathbf{nil}, s, s, 1, 1, 3)$. Павялічваем паток ўздоўж шляху $(s,1,3,t)$ на $\epsilon = \min\{3,4,3\} = 3$.
2. $Q = \{s,2,3,4,1,t\}$, $parent = (\mathbf{nil}, 3, s, 2, 2, 4)$. Павялічваем паток ўздоўж шляху $(s,2,4,t)$ на $\epsilon = \min\{8,1,5\} = 1$.
3. $Q = \{s,2,3,1,4,t\}$, $parent = (\mathbf{nil}, 3, s, 2, 1, 4)$. Павялічваем паток ўздоўж шляху $(s,2,3,1,4,t)$ на $\epsilon = \min\{7,2,3,2,4\} = 2$.
4. Шляху з s у t у графе G_f няма. Бягучы паток велічыні 6 максімальны, а $S = \{1,3,4,t\}$ — ядро мінімальнага s,t-разрэза велічыні $\delta_u(S) = 6$.

Змяненні патока на ітэрацыях алгарытма прадстаўлены ў табл. 3.1. □



Малюнак 3.2: Сетка да прыкладу 3.1

Табліца 3.1:

Іт.	f							
	$(s,1)$	$(s,2)$	$(1,3)$	$(1,4)$	$(2,3)$	$(2,4)$	$(3,t)$	$(4,t)$
0	0	0	0	0	0	0	0	0
1	3	0	3	0	0	0	3	0
2	3	1	3	0	0	1	3	1
3	3	3	1	2	2	1	3	3

3.2 Алгарытм Гольдберга і Тар'яна

Няхай дадзен перадпаток f. Будзем казаць, што перадпаток f і функцыя метак $l : V \to \mathbb{Z}_+$ *узгоднены*, калі выконваецца наступная ўмова:

$$l(t) = 0, \quad l(s) = n \text{ і } l(v) \leq l(w) + 1 \quad \text{для ўсіх } (v,w) \in E_f. \tag{3.2}$$

Змястоўны сэнс гэтага азначэння ў наступным. Няхай граф адлегласцей G_f^* атрыманы дабаўленнем дугі (s,t) да графа G_f . Азначым кошты ўсіх дуг з E_f роўнымі адзінцы, а кошт дугі (s,t) роўным n. Тады лёгка бачыць, што $l(v) \leq d_f(v,t)$, дзе $d_f(v,w)$ ёсць даўжыня карацейшага шляху ад v да w у графе G_f^* .

Тэарэма 3.2 *Паток f максімальны тады і толькі тады, калі існуе ўзгодненая з ім функцыя метак l.*

Доказ. **Неабходнасць**. Няхай f — максімальны паток. Па тэарэме 2.5 у графе G_f няма шляху з s у t. Таму функцыя метак $l(v) = d_f(v,t)$ для ўсіх $v \in V$ будзе ўзгоднена з патокам f.

Дастатковасць. Няхай f і l узгоднены. Дапусцім, што ў графе G_f існуе просты шлях $s = v_0, v_1, \ldots, v_k = t$. Так як l і f узгоднены, $k < n$, то $l(v_{i-1}) \leq$

$l(v_i) + 1$ для $1 \leq i \leq k$. Склаўшы гэтыя няроўнасці, атрымаем $l(s) - l(t) \leq k < n$, што супярэчыць умове $l(t) = 0$, $l(s) = n$. □

GT-алгарытм падтрымлівае ўзгодненыя перадпаток f і функцыю метак l, карэктуе іх, выкарыстоўваючы аперацыі *push* і *relabel*. Для апісання гэтых аперацый нам будуць патрэбны наступныя азначэнні. Мы гаворым, што вяршыня v *актыўная*, калі

$$v \notin \{s, t\} \quad \text{і} \quad e_f(v) > 0.$$

Нагадам, што перадпаток f з'яўляецца патокам, калі няма ніводнай актыўнай вяршыні. Дуга (v, w) называецца *дапушчальнай*, калі

$$(v, w) \in E_f \quad \text{і} \quad l(v) = l(w) + 1.$$

GT-алгарытм, прыведзены на мал. 3.3, пачынае работу з перадпатока f, які азначаецца наступным чынам:

$$\begin{aligned} f(s, v) &= u(s, v),\ f(v, s) = -f(s, v) \quad \text{для ўсіх дуг } (s, v) \in E, \\ f(v, w) &= 0 \quad \text{для астатніх дуг.} \end{aligned}$$

Прасцейшы выбар пачатковых метак наступны:

$$l(s) = n, \quad \text{і} \quad l(v) = 0 \quad \text{для } v \in V \setminus \{s\}.$$

А найбольш дакладны:

$$l(v) = d_f(v, t) \quad \text{для усіх } v \in V.$$

Пры апошнім выбары меткі могуць быць вылічаны за час $O(m)$, калі выкарыстаць пошук у шырыню з сцёка і з вытока ў графе G_f. Усе ацэнкі, якія мы атрымаем для складанасці алгарытма, справядлівы для любых дапушчальных пачатковых метак. Але, каб спрасціць доказ, лічым, што алгарытм пачынае з прасцейшых метак. На практыцы лепш пачынаць з найбольш дакладных значэнняў метак і перыядычна карэктаваць меткі, выкарыстоўваючы пошук у шырыню.

GT-алгарытм паслядоўна выконвае, у любым парадку, аперацыі карэктыроўкі патока і метак *push* і *relabel*, якія прадстаўлены на мал. 3.4. Калі няма актыўных вяршынь, алгарытм заканчвае работу.

Базісныя аперацыі мадыфікуюць перадпаток f і меткі d. Аперацыя *push*(v, w) павялічвае $f(v, w)$ і $e_f(w)$ на $\epsilon = \min\{e_f(v), u_f(v, w)\}$, і памяншае $f(w, v)$ і $e_f(v)$ на гэтую ж велічыню. Павелічэнне называецца *насычальным*, калі пасля яго $u_f(v, w) = 0$, і *ненасычальным* у адваротным выпадку. Аперацыя *relabel*(v) робіць метку вяршыні v роўнай найбольшаму значэнню, якое не парушае ўмовы ўзгодненасці (3.2).

Цяпер мы пяройдзем да абгрунтавання карэктнасці і аналізу эфектыўнасці алгарытма.

```
generic(G, u, s, t)
{
  // ініцыялізацыя
  for ((v, w) ∈ E) {
    f(v, w) := 0;
    if (v = s) f(s, w) := u(s, w);
    if (w = s) f(v, s) := −u(s, w);
  }
  for (w ∈ V) {
    if (w = s) l(w) := n; else l(w) := 0;
  // галоўны цыкл
  for (; існуе актыўная вяршыня; ) {
    выбраць базісную аперацыю і выканаць яе;
  }
}
```

Малюнак 3.3: Алгарытм Push and Relabel

```
push(v, w)
// Прымяненне: v актыўная і (v, w) дапусцімая
{
  ϵ := min{e_f(v), u_f(v, w)};
  f(v, w) := f(v, w) + ϵ; f(w, v) := −f(v, w);
}

relabel(v)
// Прымяненне: e_f(v) > 0 і l(w) ≥ l(v) для ўсіх (v, w) ∈ E_f(v, V)
{
  l(v) := 1 + min_{(v,w)∈E_f(v,V)} l(w);
}
```

Малюнак 3.4: Базісныя аперацыі

Лема 3.3 *Няхай f і l - узгодненыя перадпаток і функцыя метак. Калі вяршыня $v \in V \setminus \{s, t\}$ мае лішак, то да яе можна прымяніць адну з базісных аперацый.*

Доказ. Для кожнай дугі $(v, w) \in E_f(v, V)$ мы маем $l(v) \leq l(w) + 1$. Калі аперацыю *push* нельга прымяніць да вяршыні v, то для ўсіх дуг $(v, w) \in E_f(v, V)$ павінна выконвацца $l(v) < l(w)+1$, або $l(w) \geq l(v)$. А гэта значыць, што да вяршыні v прымяняльная аперацыя *relabel*. □

Лема 3.4 *На працягу ўсяго алгарытма функцыя метак не ўбывае. Больш таго, пасля аперацыі* relabel*(v) метка $l(v)$ павялічыцца не менш чым на 1.*

Доказ. Паколькі меткі змяняюцца толькі аперацыямі *relabel*, то дастаткова даказаць другое сцвярджэнне лемы. Перад прымяненнем *relabel*(v)

мы маем $l(v) \leq l(w)$ для ўсіх $(v,w) \in E_f(v,V)$. Адгэтуль вынікае, што $l(v) = 1 + \min_{(v,w)\in E_f(v,V)} l(w)$ пасля *relabel*(v). □

Лема 3.5 *На працягу ўсяго алгарытма перадпаток і функцыя метак узгоднены.*

Доказ. Доказ правядзем індукцыяй па колькасці выкананых базісных аперацыяй. У пачатку алгарытма перадпаток і меткі ўзгоднены. Дапусцім, што перадпаток f і функцыя метак l узгоднены. Пакажам, што аперацыя *relabel*(v) не парушае узгодненасці перадпатока і метак. Для дугі $(v,w) \in E_f(v,V)$ пасля выканання аперацыі *relabel*(v) гарантуецца, што $l(v) \leq l(w)+1$. Цяпер разгледзім дугу $(w,v) \in E_f(V,v)$. Па леме 3.4, калі $l(w) \leq l(v)+1$ да аперацыі, то $l(w) < l(v)+1$ пасля яе.

Цяпер разгледзім аперацыю *push*(v,w). Яна можа дабавіць дугу (w,v) да E_f і можа выдаліць дугу (v,w) з E_f. У першым выпадку мы маем $l(w) = l(v)-1$, і ўзгодненасць не парушаецца. У другім выпадку з (3.2) выдаляецца адно абмежаванне, што, зразумела, не парушае ўзгодненасці. □

Тэарэма 3.3 *Пасля заканчэння алгарытма перадпаток f з'яўляецца максімальным патокам.*

Доказ. Калі алгарытм завершыць работу, то ўсе вяршыні з $V \setminus \{s,t\}$ павінны мець нулявы лішак, таму што няма актыўных вяршынь. Таму f павінен быць патокам. А так як на працягу ўсяго алгарытма перадпаток і меткі ўзгоднены, то па тэарэме 3.2 паток f максімальны. □

Ключ да аналізу эфектыўнасці алгарытма дае наступная лема, якая паказвае, што меткі не могуць вырасці вельмі моцна.

Лема 3.6 *На працягу ўсяго алгарытма $l(v) \leq 2n-1$ для любой вяршыні $v \in V$.*

Доказ. Лема трывіяльна для $v = s$ і $v = t$. Дапусцім, што $v \in V \setminus \{s,t\}$. Так як алгарытм мяняе толькі меткі актыўных вяршынь з дапамогай аперацыі *relabel*, то дастаткова разгледзець актыўную вяршыню v. Па леме 2.1, так як для перадпатоку адзінай вяршыняй з дэфіцытам з'яўляецца вытак s, у графе G_f існуе просты шлях $v = v_0, v_1, \ldots, v_k = s$, $k \leq n-1$. Так як f і l узгоднены і $(v_{i-1}, v_i) \in E_f$, то мы маем $l(v_{i-1}) \leq l(v_i)+1$. Таму, так як $l(v_k) = n$, то мы маем

$$l(v) = l(v_0) \leq l(v_k) + k \leq n + (n-1) = 2n-1.$$

□

Лема 3.6 дазваляе ацаніць колькасць аперацый *relabel*.

Лема 3.7 *Максімальная колькасць аперацый relabel для адной вяршыні не большая, чым* $2n-1$*, а агульная іх колькасць не пераўзыходзіць* $(2n-1)(n-2) < 2n^2$.

Лема 3.8 *Колькасць насычальных аперацый push не пераўзыходзіць* nm.

Доказ. Разгледзім насычальную аперацыю $push(v, w)$. Пасля адной такой аперацыі $u_f(v, w) = 0$, і другая такая аперацыя не можа здарыцца, калі $l(w)$ не вырасце па меншай меры на 2, выканаецца аперацыя $push(w, v)$ і $l(v)$ вырасце не меньш чым на 2. Калі мы супаставім кожнай насычальнай аперацыі $push(v, w)$, за выключэннем першай, папярэдняе павелічэнне меткі $l(v)$, то атрымаем верхнюю мяжу n на колькасць такіх аперацый. □

Найбольш цікавай часткай аналізу эфектыўнасці алгарытма з'яўляецца атрыманне ацэнкі на колькасць ненасычальных аперацый *push*.

Лема 3.9 *Колькасць ненасычальных аперацый* push *не пераўзыходзіць* $2n^2m$.

Доказ. Азначым *патэнцыял* Φ для бягучага перадпатока f і функцыі метак l па формуле $\Phi = \sum_{v:v \text{ актыўная}} l(v)$. Па леме 3.6 маем $0 \le \Phi \le 2n^2$. Кожная ненасычальная аперацыя $push(v, w)$ памяншае Φ не менш чым на 1, таму што $push(v, w)$ робіць вяршыню v неактыўнай і $l(w) = l(v) - 1$. Так як $\Phi = 0$ у пачатку і пасля заканчэння алгарытма, то агульная колькасць ненасычальных аперацый *push* не пераўзыходзіць сумы павелічэнняў Φ на працягу ўсяго алгарытма. Павелічэнне меткі вяршыні v на k павялічвае Φ на k. Агульная колькасць такіх павелічэнняў не пераўзыходзіць $2n^2$. Насычальная аперацыя *push* можа павялічыць Φ самае большае на $2n-2$, а агульная колькасць такіх павелічэнняў не можа быць большай $(2n-2)nm$. Сумаванне дае верхнюю мяжу $2n^2 + (2n-2)nm \le 2n^2m$ на колькасць ненасычальны аперацый *push*. □

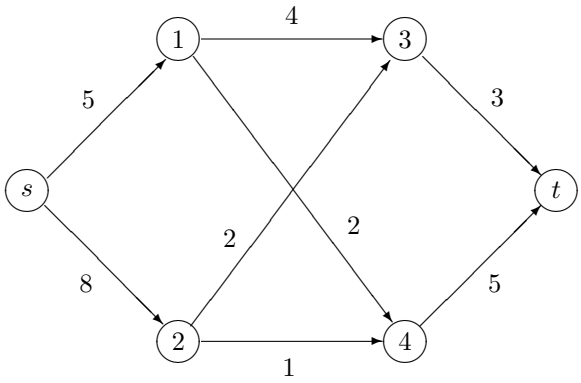
Непасрэдна з лем 3.7–3.9 атрымліваем

Тэарэма 3.4 *GT-алгарытм выконвае* $O(n^2m)$ *базісных аперацый.*

Прыклад 3.2 *Знайсці максімальны паток у сетцы на мал. 3.5 алгарытмам "Push and Relabel".*

Пачынаем рашаць задачу з перадпатока f і функцыі метак l, прадстаўленых на мал. 3.6.

Ніжэй ідуць ітэрацыі алгарытма.
1. $relabel(1)$; $l(1) = 1$.
2. $relabel(2)$; $l(2) = 1$.
3. $push(1, 3)$; $f(1, 3) = 4$; $e_f(1) = 1$, $e_f(3) = 4$.
4. $push(1, 4)$; $f(1, 4) = 1$; $e_f(1) = 0$, $e_f(4) = 1$.
5. $push(2, 3)$; $f(2, 3) = 2$; $e_f(2) = 6$, $e_f(3) = 6$.



Малюнак 3.5: Сетка да прыкладу 3.2

E	$(s,1)$	$(s,2)$	$(1,3)$	$(1,4)$	$(2,3)$	$(2,4)$	$(3,t)$	$(4,t)$
f	5	8	0	0	0	0	0	0

V	s	1	2	3	4	t
e_f	-13	5	8	0	0	0
l	6	0	0	0	0	0

Малюнак 3.6:

6. $push(2,4)$; $f(2,4)=1$; $e_f(2)=5$, $e_f(4)=2$.
7. $relabel(2)$; $l(2)=7$.
8. $relabel(3)$; $l(3)=1$.
9. $relabel(4)$; $l(4)=1$.
10. $push(2,s)$; $f(s,2)=3$; $e_f(2)=0$, $e_f(s)=-8$.
11. $push(3,t)$; $f(3,t)=3$; $e_f(3)=3$, $e_f(t)=3$.
12. $relabel(3)$; $l(3)=2$.
13. $push(4,t)$; $f(4,t)=2$; $e_f(4)=0$, $e_f(t)=5$.
14. $push(3,1)$; $f(1,3)=1$; $e_f(3)=0$, $e_f(1)=3$.
15. $relabel(1)$; $l(1)=2$.
16. $push(1,4)$; $f(1,4)=2$; $e_f(1)=2$, $e_f(4)=1$.
17. $relabel(1)$; $l(1)=3$.
18. $push(4,t)$; $f(4,t)=3$, $e_f(4)=0$, $e_f(t)=6$.
19. $push(1,3)$; $f(1,3)=3$; $e_f(1)=0$, $e_f(3)=2$.
20. $relabel(3)$; $l(3)=4$.
21. $push(3,1)$; $f(1,3)=1$; $e_f(3)=0$, $e_f(1)=2$.
22. $relabel(1)$; $l(1)=5$.
23. $push(1,3)$; $f(1,3)=3$; $e_f(1)=0$, $e_f(3)=2$.
24. $relabel(3)$; $l(3)=6$.

Табліца 3.2:

E	$(s,1)$	$(s,2)$	$(1,3)$	$(1,4)$	$(2,3)$	$(2,4)$	$(3,t)$	$(4,t)$
f	3	3	1	2	2	1	3	3

25. $push(3,1)$; $f(1,3)=1$; $e_f(3)=0$, $e_f(1)=2$.
26. $relabel(1)$; $l(1)=7$.
27. $push(1,s)$; $f(s,1)=3$; $e_f(1)=0$, $e_f(s)=-6$.

Так як бягучы перадпаток, які прадстаўлены ў табл. 3.2, з'яўляецца патокам, то гэта і ёсць максімальны паток. □

3.3 Эфектыўная рэалізацыя алгарытма Push and Relabel

Складанасць алгарытма Push and Relabel залежыць ад парадку прымянення базісных аперацый і ад дэталяў рэалізацыі. Нам спатрэбяцца некаторыя структуры дадзеных для прадстаўлення сеткі і перадпатока. Для дугі $(v,w)\in E$ неарыентаваную пару вяршынь $\{v,w\}$ называем рабром графа G. Кожнаму рабру $\{v,w\}$ ставім у адпаведнасць тры значэнні: $u(v,w)$, $u(w,v)$ і $f(v,w)=-f(w,v)$). Спіс рэбраў, інцыдэнтных вяршыні v, абазначым праз $E(v)$. Зазначым, што рабро $\{v,w\}$ належыць двум спісам — $E(v)$ і $E(w)$. У спісе $E(v)$ заўсёды вылучана *бягучае рабро* $\{v,w\}$, якое з'яўляецца першым кандыдатам для выканання аперацыі *push*. Спачатку бягучымі рэбрамі з'яўляюцца першыя рэбры спісаў. Разгледзім рэалізацыю працэдуры *generic*, у якой яе галоўны цыкл выконвае аперацыю *discharge*, прыведзеную на мал. 3.7, пакуль ёсць актыўныя вяршыні.

```
discharge (v)
Прымяненне: v актыўная
{
  Выбраць бягучае рабро {v,w} у спісе E(v);
  for (; e_f(v) ≠ 0;)
    if ((v,w) дапушчальная) push(v,w);
    else {
      замяніць (v,w) наступным рабром у E(v));
      if ((v,w) першае рабро у спісе E(v)) {
        relabel(v); return;
      }
}
```

Малюнак 3.7: Аперацыя *discharge*

Аперацыя *discharge* прымяняецца да актыўнай вяршыні v. Яна спрабуе вывесці лішак патоку з вяршыні v праз бягучае рабро $\{v, w\}$, калі аперацыя $push(v, w)$ прымяніма. Калі не, то бягучае рабро $\{v, w\}$ замяняецца наступным рабром у спісе $E(v)$ (будем лічыць, што наступным рабром для апошняга рабра спіса $E(v)$ з'яўляецца першае рабро). Калі новая бягучае рабро з'яўляецца першым рабром спіса $E(v)$, то *discharge* выконвае аперацыю $relabel(v)$ і закачвае работу. Працэдура таксама спыняецца у выпадку, калі вяршыня v стане неактыўнай ($e_f(v) = 0$).

Перайдзём да абгрунтавання карэктнасці працэдуры *discharge*.

Лема 3.10 *Аперацыя $push(v, w)$ не стварае новых дапушчальных дуг, але можа зрабіць дугу (v, w) недапушчальнай.*

Доказ. Пасля выканання $push(v, w)$ да графа G_f дадаецца толькі адна новая дуга (w, v), якая не можа стаць дапушчальнай, так як $l(w) = l(v) - 1$. Калі аперацыя $push(v, w)$ з'яўляецца насычальнай, то дуга (v, w) стане недапушчальнай. □

Лема 3.11 *Пасля выканання аперацыі $relabel(v)$ існуе хаця б адна дапушчальная дуга, якая выходзіць з v, і няма дапушчальных дуг, якія ўваходзяць у v.*

Доказ. Пасля $relabel(v)$ маем $l(v) = 1 + \min_{(v,w)\in E_f} l(w)$. Такім чынам, для вяршыні w, на якой дасягаецца мінімум, дуга (v, w) стане дапушчальнай.

Дакажам зараз, што пасля выканання $relabel(v)$ у вяршыню v не ўваходзіць ніводная дапушчальная дуга. Калі б такая дуга (w, v) існавала, то пасля $relabel(v)$ мы мелі $l(w) = l(v)+1$. Таму перад $relabel(v)$ было б $l(w) > l(v)+1$. Па азначэнню функцыі метак мы павінны заключыць, што $(w, v) \notin E_f$. А паколькі $relabel(v)$ не змяняе графа G_f, то і пасля яе выканання дугі (w, v) не будзе ў графе G_f. Атрыманая супярэчнасць завяршае доказ. □

Наступная лема паказвае, што працэдура *discharge* выконвае аперацыю *relabel* карэктна.

Лема 3.12 *Працэдура $discharge(v)$ выконвае аперацыю $relabel(v)$ тады і толькі тады, калі яна прымяніма.*

Доказ. Відавочна, што нам дастаткова паказаць, што ўсе дугі, якія выходзяць з v, недапушчальныя. *Праходам* вяршыні v назавем інтэрвал паміж дзвюмя паслядоўнымі выклікамі аперацыі $relabel(v)$ (пачатак першага праходу супадае з пачаткам алгарытму). Па завяршэнні праходу кожная дуга, якая выходзіць з v стане недапушчальнай. Так як $relabel(v)$ да канца праходу не выконваецца, то ў выніку лемаў 3.10 і 3.11 яны застаюцца недапушчальнымі і да канца праходу, г.зн. перад чарговым выклікам $relabel(v)$. □

Лема 3.13 *Версія алгарытма Push and relabel, якая грунтуецца на аперацыі* discharge*, выконваецца за час* $O(nm)$ *плюс агульны час, патрэбны для ўсіх ненасычальных аперацый* push *і на падтрымку актыўных вяршынь.*

Любое прадстаўленне мноства актыўных вяршынь, якое дазваляе выканаць улучэнне, выдаленне і доступ да актыўнай вяршыні за час $O(1)$, у сілу лем 3.8 і 3.13 дазваляе рэалізаваць алгарытм, які грунтуецца на аперацыі *discharge*, з працаёмкасцю $O(nm^2)$ (*push* можа быць рэалізавана за час $O(1)$).

Фіксуючы парадак апрацоўкі актыўных вяршынь, мы можам палепшыць эфектыўнасць алгарытма. Былі прапанаваны два натуральных парадкі. Першы — *FIFO-алгарытм*: падтрымліваць мноства актыўных вяршынь у выглядзе чаргі, выконваць аперацыю *discharge* для першай вяршыні чаргі і дадаваць новую актыўную вяршыню да канца чаргі. Другі, *метад найбольшай меткі*, выконвае аперацыю *discharge* для вяршыні з найбольшай меткай.

Апішам больш дэталёва, як рабіць выбар вяршыні з найбольшай меткай. Для гэтага падтрымліваецца масіў мноствaў B_i, $0 \le i \le 2n-1$, і індэкс b. Мноства B_i утрымлівае ўсе актыўныя вяршыні з меткай i. Яно з'яўляецца двухзвязным спісам, што патрабуе час $O(1)$ для ўлучэння і столькі ж для выдалення. Індэкс b ёсць найбольшая метка актыўнай вяршыні. Пры ініцыялізацыі, калі насычаюцца дугі, якія выходзяць з вытоку s, атрыманыя актыўныя вяршыні памяшчаюцца ў B_0, і $b = 0$. На кожнай ітэрацыі алгарытм выбірае (і выдаляе) вяршыню $v \in B_b$, выконвае $discharge(v)$, і карэктуе b. Метад спыняецца, калі b становіцца адмоўным, г. зн., калі няма актыўных вяршынь. Такая апрацоўка актыўных вяршынь, якая рэалізуе галоўны цыкл алгарытма *Push and Relabel*, прадстаўлена на мал. 3.8.

```
discharge (v)
Прымяненне: v актыўная
{
  v ← B_b;
  old_label := l(v);
  discharge(v);
  Дадаць кожную вяршыню w,якая стала актыўнай
  пасля discharge(v), да B_{l(w)}
  if (l(v) ≠ old_label) {
    b := l(v); B_b → v;   }
  else if (B_b = ∅) b := b − 1;
}
```

Малюнак 3.8: Процедура *process_vertex*

Каб зразумець, чаму працэдура *process_vertex* карэктна пералічвае b, заўважым, што $discharge(v)$ ці перапамячае v, ці робіць яе неактыўнай, але не адначасова. У першым выпадку v з'яўляецца актыўнай вяршыняй

з найбольшай меткай, таму b трэба павялічыць да значэння $l(v)$. У другім выпадку, лішак з вяршыні v перамяшчаецца ў вяршыні з меткамі $b-1$. Таму, калі мноства B_b пустое, то b трэба паменшыць на адзінку. Агульны час, які патрэбен на карэкцыю b на працягу ўсяго алгарытма, ёсць $O(n^2)$.

Вузкім месцам пры ацэнцы складанасці двух метадаў, FIFO і найбольшай меткі, з'яўляецца колькасць ненасычальных аперацыяў *push*. Мы атрымаем верхнюю мяжу $O(n^3)$ на колькасць такіх аперацій з дапамогай падзялення вылічэнняў на фазы, якія азначаюцца па рознаму для кожнага з метадаў. Для метада FIFO фаза 1 улучае аперацыі *discharge*, прымяняемых да вяршынь, якія дадаюцца ў чаргу пры ініцыялізацыі f; фаза $i+1$ ($i \geq 1$) складаецца з аперацый *discharge*, прымяняемых да вяршынь, якія дадаюцца ў чаргу на i-й фазе. Для метада найбольшай меткі фаза азначаецца максімальным інтервалам часу, на працягу якога значэнне b застаецца пастаянным.

Лема 3.14 *Колькасць фаз для алгарытмаў FIFO і найбольшай меткі не пераўзыходзіць* $4n^2$.

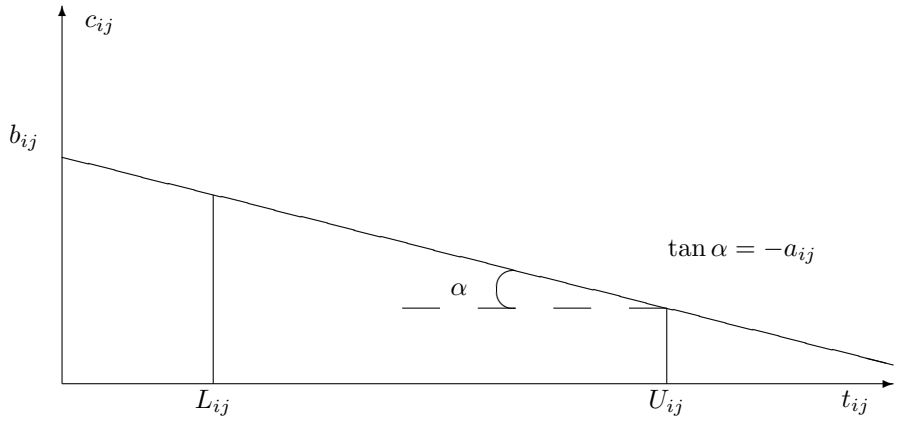
Доказ. Азначым патэнцыял бягучага перадпатока f і функцыі метак l праз

$$\Phi \stackrel{\text{def}}{=} \max_{v:\, v \text{ актыўная}} l(v);$$

$\Phi = 0$, калі няма актыўных вяршынь. Зазначым, што для алгарытма найбольшай меткі $\Phi = b$, за выключэннем моманту, калі алгарытм спыняецца. Колькасць фаз, якія выконваюць адну або больш аперацый *relabel* не пераўзыходзіць $2n^2$ (лема 3.6). Фаза, якая не выконвае *relabel*, памяншае Φ хаця б на адзінку. Пачатковае і канчатковае значэнні Φ роўны нулю. Таму колькасць фаз, якія не мяняюць метак, не большая за суму павелічэнняў Φ на працягу ўсяго алгарытма. Але Φ павялічваецца толькі пасля выканання *relabel*, прычым, павелічэнне меткі на k можа павялічыць Φ не больш чым на k. Такім чынам, сума павелічэнняў Φ на працягу ўсяго алгарытма не большая за $2n^2$, і, значыць, колькасць фаз, якія не мяняюць метак, таксама не большая за $2n^2$. □

Тэарэма 3.5 *Складанасць алгарытмаў FIFO і найбольшай меткі* $O(n^3)$.

Доказ. Абодва алгарытмы выконваюць не больш адной ненасычальнай аперацыі *push* для кожнай вяршыні на працягу адной фазы. Таму па леме 3.14 агульная колькасць ненасычальных аперацый *push* ёсць $O(n^3)$. Адкуль з лемы 3.13 атрымліваем сцвярджэнне тэарэмы. □



Малюнак 3.9: Залежнасць працягласці работ ад выдаткаў

3.4 Размеркаванне рэсурсаў у графіках праектаў

Прывядзем прыклад вельмі важнай у практыцы задачы кіравання рэсурсамі, для вырашэння якой патрабуецца прымяненне алгарытма пошуку мінімальнага разрэза, ці, што тое ж самае, максімальнага патоку.

Вельмі часта выкананне часткі, ці нават усіх работ праекта можна паскорыць, калі вылучыць дадатковую колькасць рэсурсаў. Дапусцім, што кіраўнік праекта можа ацаніць працягласць выканання работы як функцыю ад суммы грошай, выдзеленых на праект. Увядзем наступныя абазначэнні:

- t_{ij} — працягласць работы (i, j);
- L_{ij} — мінімальная працягласць работы (i, j);
- U_{ij} — максімальная ("нармальная") працягласць работы (i, j);
- $c_{ij}(t_{ij})$ — кошт выканання работы (i, j) за час t_{ij}.

Будзем лічыць, што існуе лінейная залежнасць працягласці ад затрат:

$$c_{ij}(t_{ij}) = b_{ij} - a_{ij}t_{ij}, \text{ дзе } L_{ij} \leq t_{ij} \leq U_{ij},$$

якая адлюстравана на мал. 3.9.

Для праекта, прадстаўленага сеткавым графікам $G = (N, E)$, трэба пабудаваць графік залежнасці крытычнага часу ад затрат на выкананне праекта.

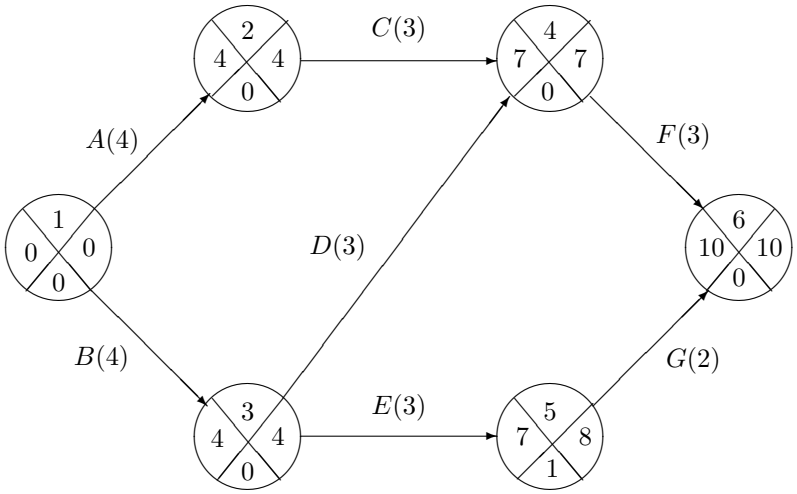
Прыклад 3.3 *Пабудаваць графік залежнасці крытычнага часу ад колькасці дадатковых рэсурсаў па дадзеным, якія прыведзены ў табл. 3.3.*

resource_ distribution(G, L, U)
{
for $((i,j) \in E)$ $t_{ij} := U_{ij}$;
$k = 0$; // k - колькасць кропак злому **for** (;;) {
знаходзім $R^s(i,j)$ для ўсіх $(i,j) \in E$;
$t_k := T_{\text{кр}}$; $C_k := \sum_{(i,j)\in E} c_{ij}\, t_{ij}$;
$E_{\text{кр}} = \{(i,j) \in E : R^s(i,j) = 0\}$;
for $((i,j) \in E_{\text{кр}})$
if $(t_{ij} > L_{ij})$ $u(i,j) := a_{ij}$;
else $u(i,j) := \infty$;
знаходзім мінімальны $1, \overline{n}$-разрэз $E_{\text{кр}}(S, V \setminus S)$
у сетцы $(G_{\text{кр}}, u, 1, n)$;
if $(\delta_u(S) = \infty)$ break;
у графе $G' = (N, E \setminus E(S, V \setminus S)$
знаходзім шлях P з 1 у n максімальнага кошту $t(P)$;
$\delta_1 := T_{\text{кр}} - t(P)$;
$\delta_2 := \min_{(i,j)\in E(P)} t_{ij} - L_{ij}$;
$\delta := \min\{\delta_1, \delta_2\}$;
for $((i,j) \in E(P))$ $t_{ij} := t_{ij} - \delta$;
$k := k + 1$;
}
}

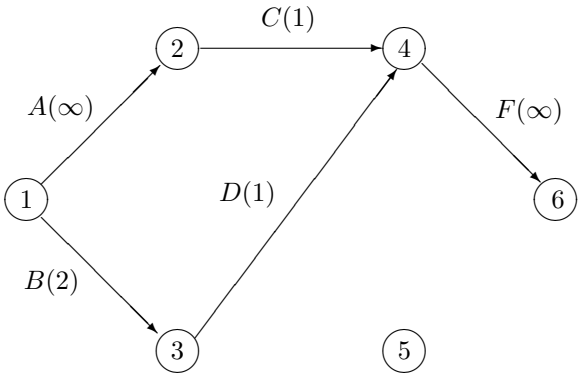
Малюнак 3.10: Алгарытм рашэння задачы размеркавання рэсурсаў

Табліца 3.3:

Работа	Працягласць U_{ij}	Працягласць L_{ij}	Непасрэдна папярэднія	Дуга	b_{ij}	a_{ij}
A	4	4	-	$(1,2)$	5	0
B	4	3	-	$(1,3)$	10	2
C	3	1	A	$(2,4)$	6	1
D	3	1	B	$(3,4)$	7	1
E	3	2	B	$(3,5)$	12	3
F	3	3	D, C	$(4,6)$	7	2
G	2	2	E	$(5,6)$	1	0



Малюнак 3.11:



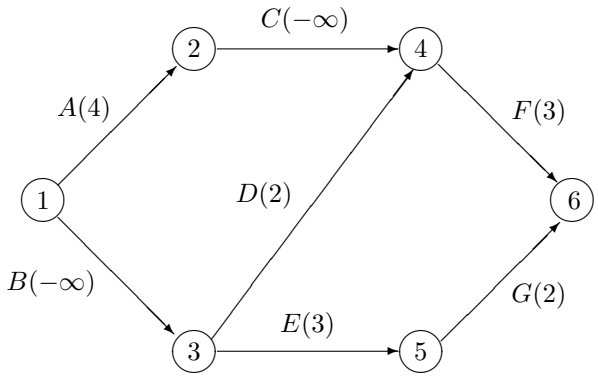
Малюнак 3.12: Граф $G^1_{\text{кр}}$

1. Сеткавы графік з вынікамі разлікаў прыведзены на мал. 3.11. $T^1_{\text{кр}} = 10$, $C^1 = 19$. Будуем граф $G^1_{\text{кр}}$ (мал. 3.12). Мінімальны разрэз у ім $E(S, V \setminus S) = \{(2,4),(3,4)\}$. Будуем граф $G^1(R)$ (мал. 3.13) і выконваем разлікі. $L^1_{\text{кр}} = 9$, $\delta_1 = T^1_{\text{кр}} - L^1_{\text{кр}} = 10 - 9 = 1$, $\delta_2 = \min\{3-1, 3-1\} = 2$, $\delta = 1$, $t_{24} = 3 - 1 = 2$, $t_{34} = 3 - 1 = 2$.

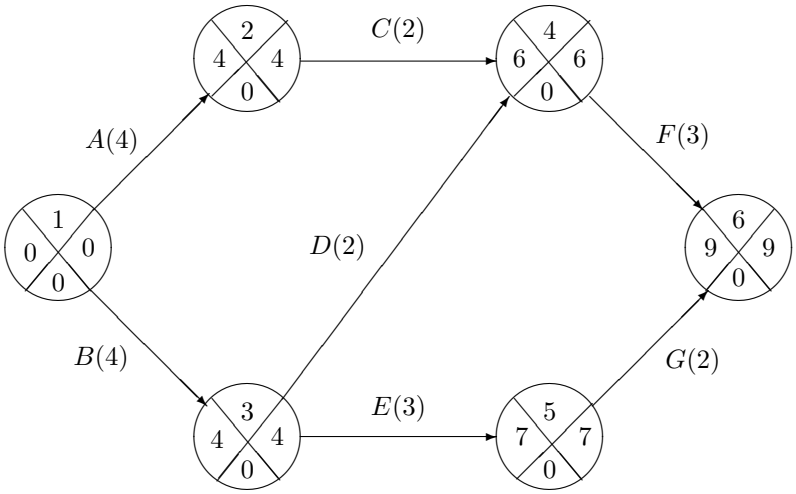
2. Выконваем разлік сеткавага графіка пры зменены х t_{ij} (мал. 3.14). Атрымліваем $T^2_{\text{кр}} = 9$, $C^2 = C^1 + 2 = 21$.

Будуем граф $G^2_{\text{кр}}$ (мал. 3.15). Мінімальны разрэз у ім $R = \{(2,4),(1,3)\}$.

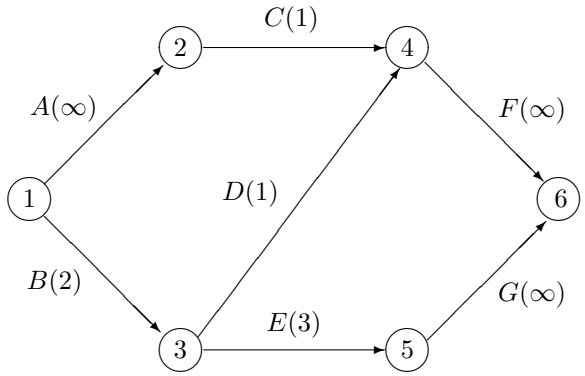
Будуем граф $G^2(R)$ (мал. 3.16) і знаходзім: $L^2_{\text{кр}} = -\infty$, $\delta_1 = \infty$, $\delta_2 =$



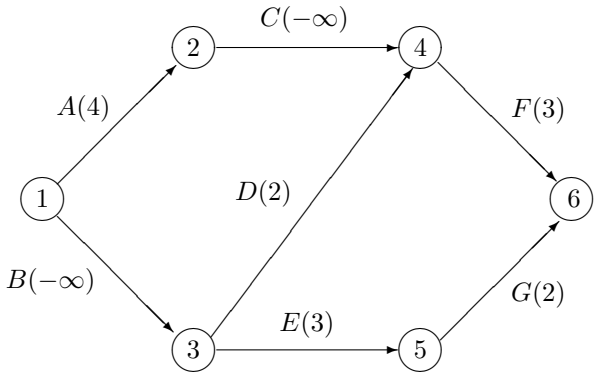
Малюнак 3.13: Граф $G^1(R)$



Малюнак 3.14:



Малюнак 3.15: Граф $G^2_{\text{кр}}$



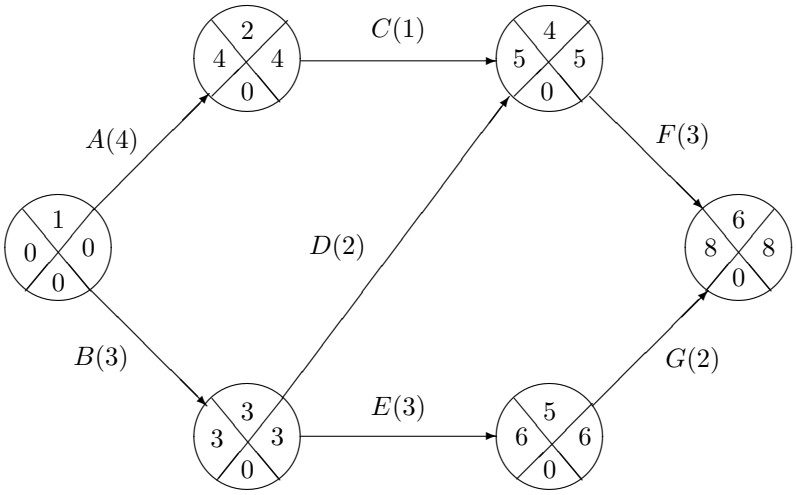
Малюнак 3.16: Граф $G^2(R)$

$\min\{3-2, 4-3\} = 1$, $\delta = 1$, $t_{24} = 2-1 = 2$, $t_{13} = 4-1 = 3$.

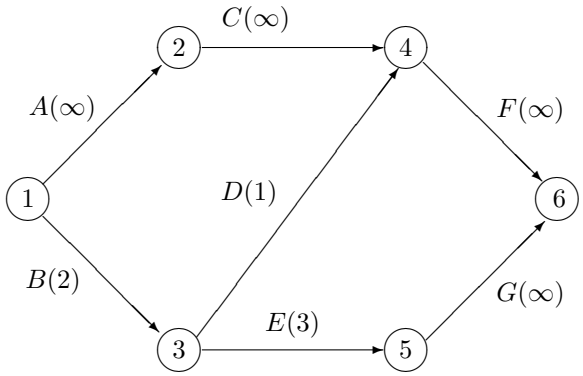
Выконваем разлік сеткавага графіка пры змененых t_{ij} (мал. 3.17). Атрымліваем $T^3_{\text{кр}} = 8$, $C^3 = C^2 + 3 = 24$.

Будуем граф $G^3_{\text{кр}}$ (мал. 3.18). Мінімальны разрэз у ім $R = \{(2,4),(3,4),(3,5)\}$ мае велічыню ∞. Работа алгарытма закончана. Графік залежнасці крытычнага часу ад дадатковых выдаткаў прадстаўлены на мал. 3.19.

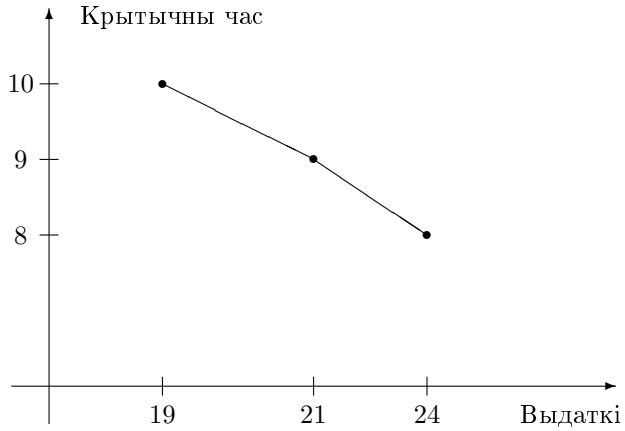
□



Малюнак 3.17:



Малюнак 3.18: граф $G^3_{\text{кр}}$



Малюнак 3.19: Графік залежнасці крытычнага часу ад выдаткаў

Глава 4

Транспартная задача

4.1 Метад адмоўных цыклаў

Непасрэдна з тэарэмы 2.2 вынікае наступны метад рашэння транспартнай задачы на сетцы (G, u, c, d), які называюць *метадам адмоўных цыклаў*. Метад пачынае працаваць з адвольнага дапушчальнага псеўдапатока f. Калі граф G_f не мае цыклаў адмоўнага кошту, то f — аптымальны псеўдапаток. Інакш, выбіраецца нейкі цыкл адмоўнага кошту, вылічваецца яго астатняя прапускная здольнасць δ, і павялічваюцца патокі на дугах гэтага цыкла на δ. Апісанне метада адмоўных цыклаў прыведзены на мал. 4.1.

Колькасць ітэрацый метада адмоўных цыклаў можа быць экспаненцыяльнай пры цэлалікавых прапускных здольнасцях і коштах. У выпадку, калі прапускныя здольнасці і кошты ірацыянальныя, метад можа і не спыніцца. На шчасце, версія алгарытма, калі на кожнай ітэрацыі псеўдапаток змяняецца ўдоўж цыклаў мінімальнага сярэдняга кошту, аказалася *строга палінаміяльнай*[1]. Гэтую версію будзем называць *метадам мінімальнага сярэдняга цыкла*.

За меру якасці псўдапатока f прымем лік $\epsilon(f)$.

Лема 4.1 *Паслядоўнасць з m ітэрацый метада мінімальнага сярэдняга цыкла скарачае $\epsilon(f)$ не менш чым у $(1 - 1/n)$ разоў.*

Доказ. Бягучы дапушчальны псеўдапаток будзем абазначаць праз f. Няхай у нейкі момант p ёсць функцыя цэн, адносна якой f з'яўляецца ϵ-аптымальнай, дзе $\epsilon = \epsilon(f)$. Падтрымліваючы ϵ і p фіксаванымі, мы даследуем як змяняецца дапушчальны граф $G_A = (V, E_A)$, дзе $E_A \stackrel{\text{def}}{=} \{(v, w) \in E_f : \ c_p(v, w) < 0\}$ пасля m мадыфікацый f. Спачатку для кожнай дугі $(v, w) \in E_A$ выконваецца $c_p(v, w) \geq -\epsilon$. Змяненне патока ўздоўж цыкла, ўсе дугі якога належаць E_A, дадае да E_f толькі дугі дадатнага прыведзенага кошту і выдаляе дакладна адну дугу з E_A. Разгледзім два выпадкі.

[1] Нагадаем, што алгарытм для вырашэння патокавай задачы з'яўляецца *строга палінаміяльным*, калі яго складанасць ацэньваецца паліномам ад n і m.

```
cycle_canceling(G, u, c, f)
{
  for ( ; у G_f існуе адмоўны цыкл ; ){
    Знайсці адмоўны цыкл C = (v_0, v_1, ..., v_{k-1}, v_k = v_0);
    δ := min_{1≤i≤k} u_f(v_{i-1}, v_i);
    for (i := 1; i ≤ k; i := i + 1) {
      f(v_{i-1}, v_i) := f(v_{i-1}, v_i) + δ;
      f(v_i, v_{i-1}) := −f(v_{i-1}, v_i);
    }
  }
}
```

Малюнак 4.1: Метад адмоўных цыклаў

1. Ні на адной з m ітэрацый цыкл мінімальнага сярэдняга кошту не мае дугі з неадмоўным прыведзеным коштам. Тады кожная з ітэрацый памяншае моц E_A і пасля не болей чым m ітэрацый мноства E_A будзе пустым, што вызначае аптымальнасць бягучага псеўдапатока f. У гэтым выпадку лема выконваецца, так як $\epsilon(f) = 0$.

2. На нейкай ітэрацыі цыкл мінімальнага сярэдняга кошту мае дугу неадмоўнага прыведзенага кошту. Няхай Y ёсць першы такі цыкл. Прыведзены кошт кожнай дугі з Y не меншы $-\epsilon$, адна дуга мае неадмоўны прыведзены кошт, а колькасць дуг цыкла Y не большая за n. Таму сярэдні кошт Y не меншы $-(1 - 1/n)\epsilon$. Такім чынам, перад ітэрацыяй, якая апрацоўвае Y, па тэарэме 2.12, $\epsilon(f) \leq (1 - 1/n)\epsilon$. Так як $\epsilon(f)$ ніколі не павялічваецца, то і ў гэтым выпадку лема справядліва. □

Калі выкарыстаць лему 4.1, то можна атрымаць палінаміяльную мяжу на колькасць ітэрацый метада мінімальнага сярэдняга цыкла, пры ўмове, што ўсе кошты цэлыя.

Тэарэма 4.1 *Калі ўсе кошты дуг цэлалікавыя, то метад мінімальнага сярэдняга цыкла спыняецца пасля* $O(nm \log(n\|c\|_\infty))$ *ітэрацый.*

Доказ. Тэарэма вынікае з лемы 4.1, тэарэмы 2.10 і таго, што любы паток з'яўляецца $\|c\|_\infty$-аптымальнай. □

Доказ наступнай тэарэмы, якая ўсталёўвае строга палінаміяльную ацэнку працаёмкасці метада мінімальнага сярэдняга цыкла, выкарыстоўвае наступную няроўнасць:

$$(1 - 1/n)^{n(\ln n+1)} \leq 1/2n \text{ для } n \geq 2.$$

Тэарэма 4.2 *Для любых сапраўдных коштаў дуг колькасць ітэрацый метада мінімальнага сярэдняга цыкла не пераўзыходзіць* $O(nm^2 \log n)$.

Доказ. Няхай $k = m(n\lceil \ln n \rceil + 1)$. Падзелім усе ітэрацыі на групы па k паслядоўных ітэрацый у групе. Пакажам, што кожная з груп фіксуе паток па меньшай меры на адной дузе (v, w), г. зн., што на ўсіх ітэрацыях, якія ідуць пасля ітэрацый гэтай групы, $f(v, w)$ не мяняецца. Адсюль і будзе вынікаць сцвярджэнне тэарэмы.

Разгледзім нейкую групу ітэрацый. Няхай f ёсць псеўдапаток перад першай ітерацыяй групы, f' — псеўдапаток пасля апошняй ітэрацыі групы, а p' ёсць функцыя цэн, для якой f' задавальняе ўмове $\epsilon(f')$-аптымальнасці. Няхай на першай ітэрацыі групы паток мяняецца ўздоўж цыкла Y. Па леме 4.1 і з выбару k маем

$$\epsilon(f') \leq \epsilon(f)(1 - 1/n)^{n\lceil \ln n \rceil + 1} \leq \epsilon(f)/2n.$$

Так як сярэдні кошт цыкла Y роўны $-\epsilon(f)$, то для нейкай яго дугі, скажам (v, w), павінна выконвацца

$$c_{p'}(v, w) \leq -\epsilon(f) \leq -2n\epsilon(f').$$

Згодна з вынікам 2.1, паток па дузе (v, w) не будзе мяняцца пасля заканчэння ўсіх ітэрацый дадзенай групы. Акрамя таго, $f(v, w)$ змяніўся на першай ітэрацыі групы. Такім чынам, кожная група фіксуе паток па меньшай меры на адной дузе. □

Непасрэдна з тэарэм 4.1 і 4.2 маем

Тэарэма 4.3 *Складанасць метада мінімальнага сярэдняга цыкла ёсць* $O(n^2m^3 \log n)$ *для сапраўдных коштаў дуг і* $O(n^2m^2 \min\{\log(n\|c\|_\infty), m \log n\})$*, калі ўсе коштыцэлыя.*

4.2 Сеткавы сімплекс-метад

Разгледзім транспартную задачу на сетцы (G, u, c, d). Зафіксуем адвольную вяршыню $s \in V$. Няхай f — нейкі дапушчальны псеўдапаток, T — ардрэва з корнем s у графе G. У кантексце сеткавага сімплекс-метада дрэва T называецца *базісным*. Выличым функцыю p адлегласцей дрэва T. Па азначэнню функцыі адлегласцей $c_p(v, w) = 0$ для ўсіх дуг $(v, w) \in E(T)$, а таксама і для ўсіх адваротных ім дуг. Магчымы два выпадкі:

1) $c_p(v, w) \geq 0$ для ўсіх дуг $(v, w) \in E_f$;

2) існуе дуга $(v, w) \in E_f$ такая, што $c_p(v, w) < 0$.

У першым выпадку па тэарэме 2.3 паток f з'яўляецца аптымальным. Разгледзім другі выпадак. Няхай $P = (w = v_0, v_1, \ldots, v_k = v)$ ёсць адзіны (неарыентаваны) шлях ў дрэве T з w у v. Так як граф G сіметрычны, то P таксама з'яўляецца арыентаваным шляхам у графе G. Яго кошт роўны $p(v) - p(w)$. Тады

$$p(v) - p(w) + c(v, w) = c_p(v, w) < 0$$

```
net_simplex(G, u, c, f) // f — дапушчальны псеўдапаток
{
  выбраць пакрывальнае ардрэва T графа G з корнем s ∈ V;
  знайсці функцыю p адлегласцей ардрэва T;
  for ( ; ((v, w) := check_opt(f, p)) ≠ nil; ) {
    У графе T + (v, w) вылучыць цыкл (v_0 = v, w = v_1, v_2, ..., v_k = v);
    δ := min_{0≤i<k} u_f(v_i, v_{i+1});
    Выбраць дугу (x, y) = (v_j, v_{j+1}), такую, што δ = u_f(v_j, v_{j+1});
    if ((x, y) ≠ (v, w)) {
      мадыфікаваць ардрэва T наступным чынам:
      a) выдаліць з T дугу (x, y) і дадаць дугу (v, w),
      b) пераарыентаваць дугі адрыманага дрэва у напрамку
           ад корня s да лісцяў;
      c) вылічыць функцыю p адлегласцей новага адрэва T;
    }
    for (i := 0; i < k; i := i + 1)
      f(v_i, v_{i+1}) := f(v_i, v_{i+1}) + δ;
      f(v_{i+1}, v_i) := −f(v_i, v_{i+1});
    }
  }
}
```

Малюнак 4.2: Сеткавы сімплекс-метад

ёсць кошт цыкла $(v_0, v_1, \ldots, v_k, v_{k+1} = v_0)$. Няхай

$$\delta = \min\{u_f(v_i, v_{i+1}) : 0 \le i \le k\}.$$

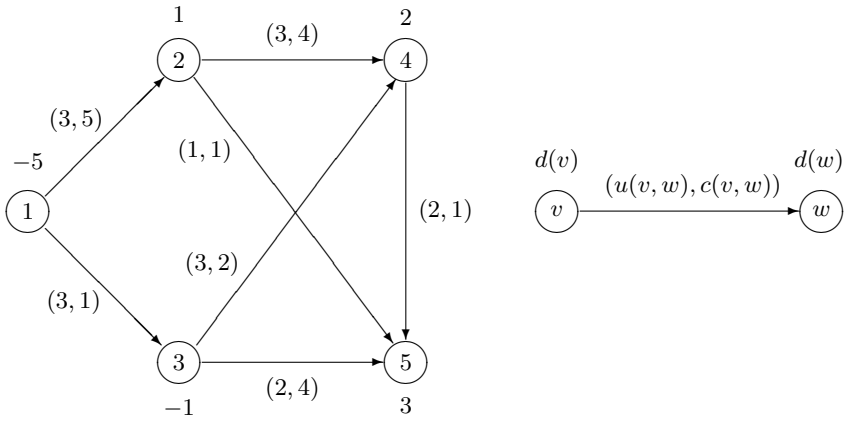
Дуга $(x, y) = (v_j, v_{j+1})$ шляху P, такая, што $u_f(x, y) = \delta$, называецца *блакіруючай*. Заўважым, што δ можа быць роўным нулю. Зменім псеўдапаток f па правілу:

$$\begin{aligned} f(v_i, v_{i+1}) &:= f(v_i, v_{i+1}) + \delta, \\ f(v_{i+1}, v_i) &:= -f(v_i, v_{i+1}), \quad i = 0, \ldots, k, \\ f(x, y) &:= f(x, y) \text{ для астатніх дуг.} \end{aligned}$$

Пры гэтым кошт псеўдапатока паменшыцца на $-\delta c_p(v, w)$. Такая стратэгія паляпшэння дапушчальнага псеўдапатока ляжыць ў аснове *сеткавага сімплекс-метада*, вядомага таксама пад назвай *метад патэнцыялаў*. Працэдура *net_simplex*, якая рэалізуе сеткавы сімплекс-метад, прадстаўлена на мал. 4.2.

Прыклад 4.1 *Сеткавым сімплекс-метадам вырашыць транспартную задачу ў сетцы, якая прыведзена на мал. 4.3*

Каб знайсці пачатковы дапушчальны псеўдапаток, пашырым сетку (гл. мал. 4.4), дадаўшы дадатковую вяршыню 6 і 5 дуг $(2, 6)$, $(4, 6)$, $(5, 6)$ і $(6, 1)$,



Малюнак 4.3: Сетка да прыкладу 4.1

(6, 3). па адной дузе для кожнай вяршыні графа. Кошты ўсіх новых дуг азначым роўнымі 10 (∞), а прапускныя здольнасці і патокі на дугах $(6, v)$ і $(v, 6)$ — роўнымі $|d(v)|$. Рашэнне пачынаем з ардрэва T, якое прадстаўлена на мал. 4.5) (лікі на дугах — гэта іх кошты, а лікі ля вяршынь — гэта іх цэны). Ніжэй ідуць ітэрацыі метада. Змяненні дугавых патокаў на ітэрацыях адлюстраваны ў табл. 4.1.

1. $c_p(1,2) = -10 + 5 - 10 = -15 < 0$, $(v, w) = (1, 2)$, $C = (1, 2, 6, 1)$, $\delta = \min\{3, 1, 1\} = 1$, $(x, y) = (2, 6)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.6.a).

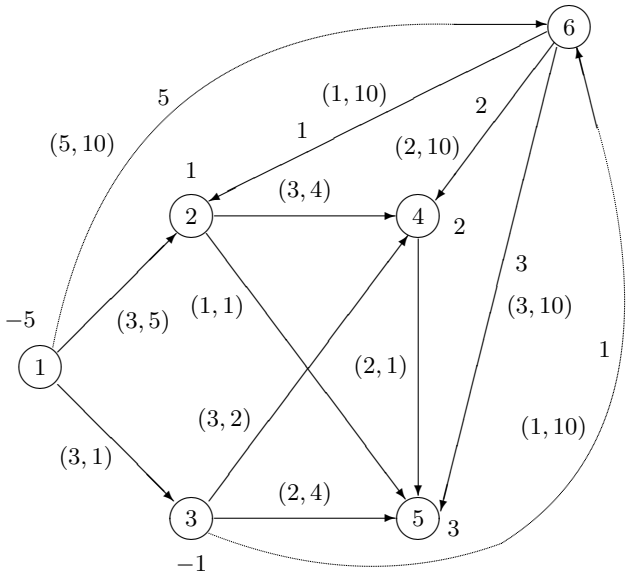
2. $c_p(1,3) = -10 + 1 + 10 = 1$, $c_p(2,4) = -5 + 4 - 10 = -11$, $(v, w) = (2, 4)$, $C = (2, 4, 6, 1, 2)$, $\delta = \min\{3, 2, 4, 2\} = 2$, $(x, y) = (1, 2)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.6.b).

3. $c_p(2,1) = 6 - 5 + 10 = 11$, $c_p(1,3) = -10 + 1 + 10 = 1$, $c_p(2,5) = 6 + 1 - 10 = -3$, $(v, w) = (2, 5)$, $C = (2, 5, 6, 4, 2)$, $\delta = \min\{1, 3, 6, 2\} = 1$, $(x, y) = (2, 5)$. Дадзеная ітэрацыя не мяняе базіснае дрэва.

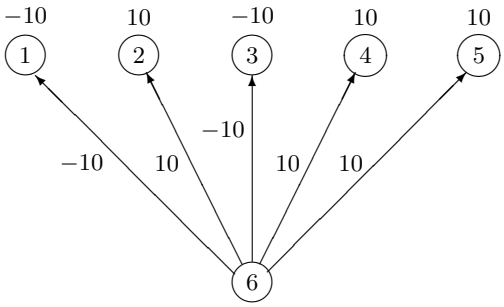
4. $c_p(3,4) = -10 + 2 - 10 = -18$, $(v, w) = (3, 4)$, $C = (3, 4, 6, 3)$, $\delta = \min\{3, 1, 1\}$, $(x, y) = (4, 6)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.7.a).

5. $c_p(3,5) = -10 + 4 - 10 = -16$, $(v, w) = (3, 5)$, $C = (3, 5, 6, 3)$, $\delta = \min\{2, 2, 0\} = 0$, $(x, y) = (6, 3)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.7.b).

6. $c_p(1,3) = -10 + 1 - 6 = -15$, $(v, w) = (1, 3)$, $C = (1, 3, 5, 6, 1)$, $\delta = \min\{3, 2, 2, 2\} = 2$, $(x, y) = (3, 5)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.8.a).



Малюнак 4.4: Пашыраная сетка да прыкладу 4.1

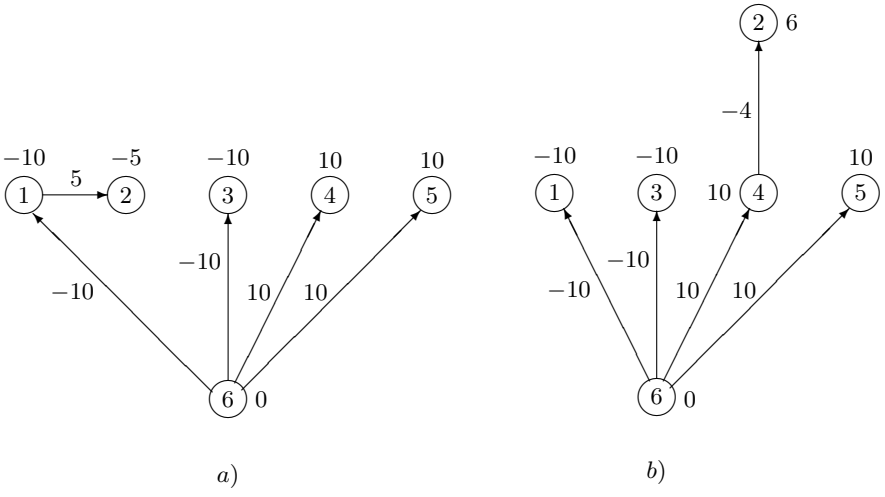


Малюнак 4.5: Пачатковае базіснае дрэва да прыкладу 4.1

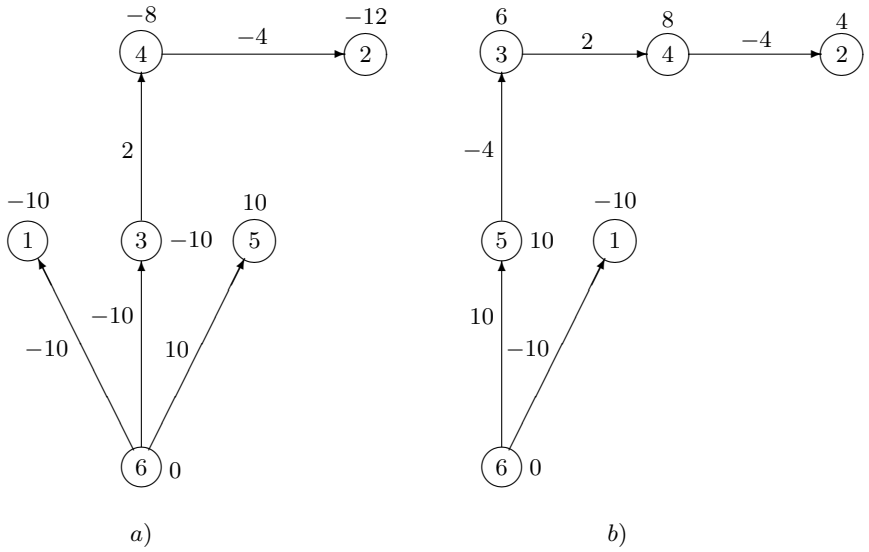
Табліца 4.1: Псеўдапатокі на розных ітэрацыях

	(1,2)	(1,3)	(2,4)	(2,5)	(3,5)	(3,4)	(4,5)
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	3	0	2	0	0	0	0
3	3	0	1	1	0	0	0
4,5	3	0	1	1	0	1	0
6	3	2	1	1	2	1	0
7,8	2	3	0	1	2	2	0
9	2	3	0	1	3	1	0

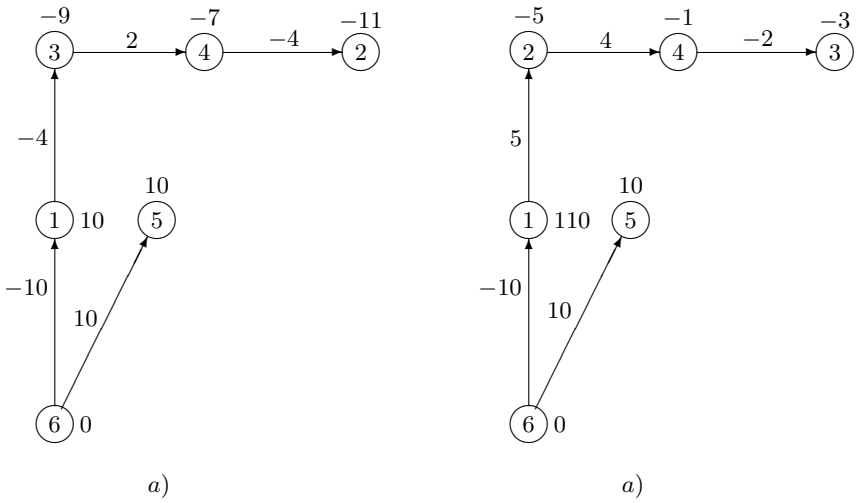
	(1,6)	(6,2)	(3,6)	(6,4)	(6,5)
0	5	1	1	2	3
1	4	0	1	2	3
2	2	0	1	0	3
3	2	0	1	1	2
4,5	2	0	0	0	2
6	0	0	0	0	0
7,8	0	0	0	0	0
9	0	0	0	0	0



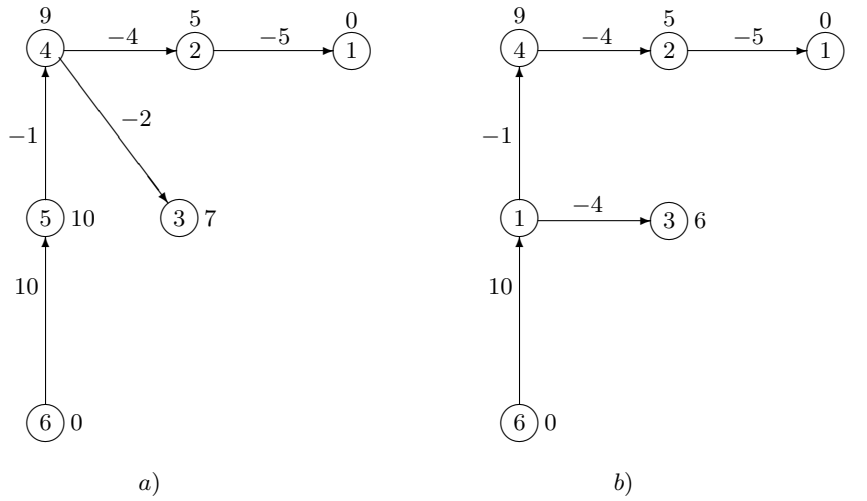
Малюнак 4.6: Базісныя дрэвы пасля ітэрацый 1,2 і 3



Малюнак 4.7: Базісныя дрэвы пасля ітэрацый 4,5



Малюнак 4.8: Базісныя дрэвы пасля ітэрацый 6,7



Малюнак 4.9: Базісныя дрэвы пасля ітэрацый 8,9

7. $c_p(2,1) = -11-5+10 = -6$, $(v,w) = (2,1)$, $C = (2,1,3,4,2)$, $\delta = \min\{3,1,2,1\} = 2$, $(x,y) = (1,3)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.8.b).

8. $c_p(3,1) = -3-1+10 = 6$, $c_p(5,2) = 10-1+5 = 14$, $c_p(5,3) = 10-4+3 = 9$, $c_p(4,5) = -1+1-10 = -10$, $(v,w) = (4,5)$, $C = (4,5,6,1,2,4)$, $\delta = \min\{2,0,6,1,3\} = 0$, $(x,y) = (5,6)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.9.a).

9. $c_p(3,1) = -3-1+10 = 6$, $c_p(5,2) = 10-1-5 = 4$, $c_p(5,3) = 10-4-7 = -1$, $(v,w) = (5,3)$, $C = (3,5,4,3)$, $\delta = \min\{2,1,2\} = 1$, $(x,y) = (3,4)$. Новае дрэва і функцыя цэн прадстаўлены на мал. 4.9.b).

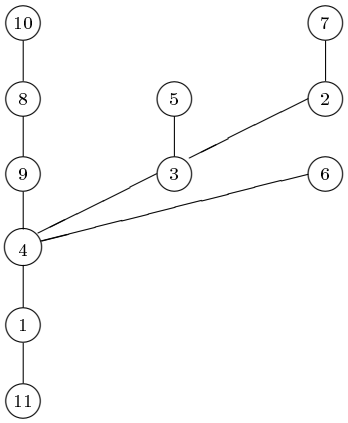
□

4.2.1 Структуры дадзеных для прадстаўлення базіснага дрэва

Базіснае дрэва зручна прадстаўляць трыма спісамі *parent*, *rank* і *next*, элементы якіх наступныя:

- $parent[v]$ — продак вузла v, перадапошняя вяршыня на шляху ад корня да v[2];

[2] Звычайна, на практыцы зручней лічыць, што $parent[v]$ ёсць апошняе рабро на шляху ад корня да v.



Малюнак 4.10: Прыклад базіснага дрэва

- $rank[v]$ — узровень вяршыні v ў дрэве (колькасць рэбраў на шляху ад корня да v);
- $next[v]$ — наступная за v вяршыня дрэва.

Спіс $next$ называюць *спісам прамога абыходу дрэва*. Для дрэва, прадстаўленага на малюнку 4.10, спісы $parent$, $next$, $rank$ маюць наступны выгляд:

v	1	2	3	4	5	6	7	8	9	10	11
$parent$	11	3	4	1	3	4	2	9	4	8	**nil**
$next$	4	7	5	9	2	11	6	10	8	3	1
$rank$	1	4	3	2	4	3	7	4	3	5	0

Для эфектыўнай рэалізацыі сеткавага сімплекс-метада трэба ўмець хутка выконваюцца наступныя аперацыі:

- вылічваць функцыю адлегласцей дрэва;
- знаходзіць адзіны шлях паміж любымі дзвумя вяршынямі;
- выдаляць рабро з дрэва;
- дабаўляць рабро да лесу дрэваў.

Як вылічыць функцыю адлегласцей дрэва

Вельмі проста, выкарыстоўваючы ўказальнікі $next$:

```
l p(s) := 0;
for (w := next(s); w ≠ s; w := next(w)) {
    p(w) := p(parent(w)) + c(parent(w), w);
}

l Δ := rank(j); rank(j) := 0;
for (q := next(v := j); rank(q) > Δ; q := next(v := q))
    rank(q) := rank(q) − Δ;
```

Выдаленне рабра з дрэва

Дапусцім, што трэба выдаліць рабро (i, j). Няхай v ёсць апошні прадпісаны паслядоўнік у дрэве з корнем у вузле j. Вяршыню v мы атрымаем пасля завяршэння наступнага фрагмента:

У гэтым фрагменце мы таксама паменшылі рангі ўсіх вяршынь паддрэва з корнем j на велічыню $\Delta = rank(j)$.

Пасля завяршэння наступнага цыкла

$$\textbf{for}(w := next(i);\ w \neq j;\ w := next(w))$$

$next(w) = j$. Калі вяршыні v, q і w знойдзены, для карэктыроўкі спісаў $next$ і $rank$ патрэбна выканаць толькі тры наступныя аперацыі:

$$next(w) := next(v);\ next(v) := j;\ parent(j) := \textbf{nil}.$$

Напрыклад, пры выдаленні рабра $(i, j) = (3, 4)$ з дрэва на мал. 4.10 атрымліваем $v = 7$, $w = 10$. Пасля выдалення спісы $parent$, $rank$ і $next$ маюць наступны выгляд:

v	1	2	3	4	5	6	7	8	9	10	11
$parent$	11	3	**nil**	1	3	4	2	9	4	8	**nil**
$next$	4	7	5	9	2	11	3	10	8	6	1
$rank$	1	1	0	2	1	3	2	4	3	5	0

Дабаўленне рабра да лесу дрэваў

Няхай трэба дабавіць рабро (i, j) да лесу дрэваў, прадстаўленых спісамі $parent$, $rank$ і $next$.

Спачатку разгледзім выпадак, калі вяршыня j з'яўляецца коранем аднаго з дрэў, г. зн., што $parent(j) = \textbf{nil}$. Пры карэктыроўцы спіса $next$ неабходна ўлічыць, што вяршыня j стане новым паслядоўнікам вяршыні i, а стары паслядоўнік вяршыні i — новым паслядоўнікам вяршыні v, якая была апошнім паслядоўнікам у паддрэве з коранем у вяршыні j. Вяршыню v мы атрымаем пасля завяршэння наступнага фрагмента:

```
Δ := rank(j) − rank(i) + 1; rank(j) := rank(j) + Δ;
for (v := next(j); v ≠ j; v := next(v))
    rank(q) := rank(q) + Δ;
```

У гэтым фрагменце мы таксама павялічылі рангі ўсіх вяршынь паддрэва з корнем j на велічыню $\Delta = rank(j) - rank(i) + 1$. Пасля таго, як вяршыня v знойдзена, неабходна ўвесці наступнгыя змяненні ў спісы $next$ і $parent$:

$$next(v) := next(i);\ next(i) := j;\ parent(j) := i.$$

Цяпер разгледзім выпадак, калі j не з'яўляецца корнем дрэва. Няхай $x = parent(j)$. Выдаляем рабро (x, j), а затым дабаўляем рабро (i, j). Рэкурсіўна прымяняем апісаную вышэй канструкцыю для дабаўлення дугі (j, x).

4.2.2 Строга дапушчальныя пакрывальныя дрэвы і складанасць сеткавага сімплекс-метада

Сеткавы сімплекс-метад не абавязкова спыняецца пасля концай колькасці ітэрацый, калі не накладваць ніякіх дадатковых абмежаванняў на выбар дуг, дабаўляемых і выдаляемых з дрэва. Як і для агульнай задачы ЛП, існуюць прыклады транспартных задач (нават малых памераў), на якіх сеткавы сімплекс-метад *зацыкліваецца*. Акрамя таго, у адрозненне ад задачы ЛП, амаль усе патокавыя задачы выраджаныя. Практыка прымянення сеткавага сімплекс-метада сведчыць, што да 90% яго ітэрацый з'яўляюцца выраджанымі. Як мы пакажам ніжэй, калі сеткавы сімплекс-метад падтрымлівае пакрывальнае дрэва спецыяльнага тыпу, то ён становіцца концым.

Пакрывальнае дрэва T у графе $\hat{G}$ называецца *строга дапушчальным* для псеўдапатока f, калі пры арыентацыі яго рэбраў у напрамку ад корня да лісцяў атрымліваецца ардрэва $\vec{T}$ ў графе G_f.

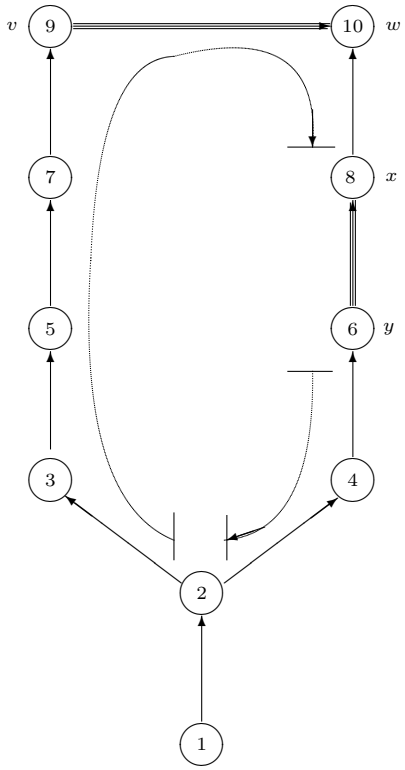
Дапусцім, што сеткавы сімплекс-метад пачынае са строга дапушчальнага пакрывальнага дрэва. Няцяжка праверыць, што такім з'яўляецца штучнае дрэва, якое мы будавалі ў прыкладзе 4.1. Дапусцім цяпер, што на нейкай ітэрацыі мы маем строга дапушчальнае пакрывальнае дрэва T і да яго збіраемся дабаўляць рабро (v, w). Няхай C адзіны цыкл у графе $T+(v, w)$, z — вяршыня цыкла C, бліжэйшая да корня у дрэве T. Як заўсёды, выбіраем арыентацыю C, каб праходзіць дугу (v, w) у прамым напрамку. Калі C мае толькі адну блакіруючую дугу, то яе і выводзім з базіса. Калі ж маецца некалькі блакіруючых дуг, то мы карыстаемся наступным правілам.

> **Правіла выбару выдаляемай дугі.** *Выбраць першую блакіруючую дугу (x, y) арцыкла C пры яго абыходзе, які пачынаецца з вяршыні z.*

Гэтае правіла праілюстравана на мал. 4.11.

Лема 4.2 *Пасля выканання ітэрацыі сеткавага сімплекс-метада з дабаўленнем дугі (v, w) і выдаленнем дугі (x, y) дрэва $T' \stackrel{\text{def}}{=} T - (x, y) + (v, w)$ з'яўляецца строга дапушчальным.*

Доказ. Няхай f і f' ёсць дапушчальныя псеўдапатокі адпаведна да і пасля ітэрацыі. Абазначым праз P_1 шлях па цыклу C ад z да x, а праз P_2



Малюнак 4.11: Выбар выдаляемай дугі

— шлях уздоўж C ад y да z (гл. мал. 4.11). Выбярэм адвольную дугу (i, j) ардрэва T' (арыентацыя ад лісцяў да корня). Разгледзім тры магчымых выпадкі.

1. Дуга (i, j) належыць шляху P_1. Правіла выбара дугі (x, y) гарантуе, што ўсе дугі шляху P_1 маюць ненулявыя астатнія прапускныя здольнасці. Таму $(i, j) \in E_f$.

2. Дуга (j, i) ляжыць на шляху P_2. Калі ітэрацыя была нявыраджанай ($\delta > 0$), то тады

$$f'(i, j) = f(i, j) - \delta < f(i, j) \leq u(i, j).$$

У выпадку, калі $\delta = 0$, то, так як дрэва T з'яўляецца строга дапушчальным, на яго шляху ад z да v няма дуг нулявой астатняй прапускной здольнасці. Таму $f'(i, j) = f(i, j) < u(i, j)$.

3. Рабро (i, j) не належыць цыклу C. Тады $f'(i, j) = f(i, j) < u(i, j)$. □

Тэарэма 4.4 *Калі ўсе параметры транспартнай сеткі цэлыя, то сеткавы сімплекс-метад, які ўвесь час падтрымлівае дрэва строга дапушчальным, з'яўляецца псеўдапалінаміяльным*[3].

Доказ. Без абмежавання агульнасці, можна лічыць, што значэнне функцыі мэты абмежавана (гл. тэарэму 2.4). Зазначым, што на працягу выкання алгарытма псеўдапаток і цэны таксама цэлыя. Так як кожная нявыраджаная ітэрацыя памяншае значэнне функцыі мэты не меньш чым на 1, то такіх ітэрацый не можа быць больш чым $m\|c\|_\infty\|u\|_\infty$. Пакажам цяпер, што колькасць паслядоўных выраджаных ітэрацый заўсёды канцая. Так як да дрэва T заўсёды дабаўляецца дуга (v,w) адмоўнага прыведзенага кошту $c_p(v,w) \le -1$, а прыведзеныя кошт ўсіх дуг (у абодвух накірунках) дрэва роўны нулю, то кожная ітэрацыя павялічвае цэны ўсіх вяршынь паддрэва дрэва T з коранем w на $c_p(v,w) \le -1$, то сума цэн ўсіх вяршынь дрэва строга ўбывае. А так як сума цэн вяршынь не можа быць меншай чым $-n\|c\|_\infty$, то колькасць выраджаных ітэрацый не пераўзыходзіць $n\|c\|_\infty$. □

Заўвага. Дапушчальны псеўдапаток f называецца *базісным*, калі існуе пакрывальнае ардрэва T графа G з коранем у нейкай вяршыні $s \in V$ такое, што ўмова $(v,w),(w,v) \in E_f$ цягне за сабой (v,w) або $(w,v) \in E(T)$. Відавочна, што ўласцівасць псеўдапатоку быць базісным з'яўляецца інварыянтам сеткавага сімплекс-метада, г. зн., калі метад пачынае з базіснага псеўдапатоку, то і ўсе пабудаваныя ім псеўдапатокі з'яўляюцца базіснымі.

[3] Нагадаем, што алгарытм для вырашэння транспартнай задачы на сетцы (G,u,c,d) называецца *псеўдапалінаміяльным*, калі яго складанасць абмежавана зверху паліномам ад n, m, $\|c\|_\infty$, $\|u\|_\infty$ і $\|d\|_\infty$.

Глава 5

Патокі з выйгрышамі і пройгрышамі

Да гэтай пары мы разглядалі патокавыя мадэлі, якія задавальнялі, часам не яўна, *умове захавання патока на дугах сеткі*: колькі патока выцякла з вяршыні v па дузе (v, w) столькі ж уцячэ ў вяршыню w. Аднак існуюць практычныя сітуацыі, у якіх гэты прынцып не выконваецца. Напрыклад, магчымы страты пры транспартыроўцы прадукта. Іншыя прыклады будуць прыведзены пазней.

5.1 Абагульненыя патокі

Абагульненай транспартнай сеткай называецца пяцёрка (G, u, c, γ, d), дзе, $G = (V, E)$ — сіметрычны арграф, $u : E \to \mathbb{R}$ — функцыя прапускных здольнасцей дуг, $c : E \to \mathbb{R}$ — антысіметрычная функцыя кошту дуг, $d : V \to \mathbb{R}$ — функцыя попыту. Зазначым, што цяпер мы не патрабуем выканання ўмовы $\sum_{v \in V} d(v) = 0$. У дадатак да параметраў стандартнай транспартнай задачы ўводзіцца новая функцыя $\gamma : E \to \mathbb{R}$ *выйгрышаў* дуг. Гэта значыць, што калі з вяршыні v па дузе (v, w) выцекла $f(v, w)$ адзінак патока, то ў вяршыню w уцячэ $\gamma(v, w)f(v, w)$ адзінак. Будзем лічыць, што функцыя выйгрышаў з'яўляецца *лагарыфмічна антысіметрычнай*:

$$\gamma(v, w) = \frac{1}{\gamma(w, v)} \quad \text{для ўсіх } (v, w) \in E. \tag{5.1}$$

*Абагульнены псеўдапатокёсць функцыя $g : E \to \mathbb{R}$, для якой выконваюцца наступныя ўмовы:

$$g(v, w) \leq u(v, w) \quad \text{для ўсіх } (v, w) \in E, \tag{5.2}$$

$$g(v, w) = -\gamma(w, v)g(w, v) \quad \text{для ўсіх } (v, w) \in E. \tag{5.3}$$

Тут няроўнасці (5.2) задаюць абмежаванні на прапускныя здольнсці дуг, а абмежаванні (5.3) называюць *умовай антысіметрыі для абагульненых*

патокаў. *Лішак* $e_g^\gamma(v)$ вяршыні v для абагульненага псеўдапатока g азначаецца па правілу:

$$e_g^\gamma(v) \stackrel{\text{def}}{=} -\sum_{(v,w)\in E} g(v,w). \tag{5.4}$$

Як і раней, мы будзем казаць, што вяршыня v мае *лішак*, калі $e_g^\gamma(v) > 0$, і мае *дэфіцыт*, калі $e_g^\gamma(v) < 0$.

Абагульнены псеўдапаток g будзем называць *дапушчальным*, калі ён задавальняе наступнай умове захавання:

$$e_g^\gamma(v) = d(v) \quad \text{для ўсіх } v \in V. \tag{5.5}$$

Калі ўмова (5.5) парушаецца ў вяршыні v, то будзем казаць, што ў гэтай вяршыні назіраецца *дысбаланс*.

5.1.1 Тэарэма аб дэкампазіцыі

Азначым *выйгрыш* шляху $P = (s = v_0, v_1, \ldots, v_k = t)$ па формуле:

$$\gamma(P) \stackrel{\text{def}}{=} \prod_{i=1}^{k} \gamma(v_i, v_{i-1}). \tag{5.6}$$

Калі з вяршыні s выцячэ адна адзінка прадукта, то у вяршыню t прыцячэ $\gamma(P)$ адзінак. Наступныя азначэнні вельмі важныя для разумення ідэй, на якіх будуюцца метады рашэння задач аб абагульненых патоках. Цыкл C у графе G называецца *генеруючым*, калі $\gamma(C) > 1$. Калі $\gamma(C) < 1$, то C — *абсарбуючы* цыкл. Калі ж $\gamma(C) = 1$, то C — *нейтральны* цыкл.

Па аналогіі са звычайным псеўдапатокам кожны абагульнены псеўдапаток можна прадставіць як суму невялікай ($\leq m$) колькасці "элементарных абагульненых псеўдапатокаў". Як і раней, кожны з гэтых прымітыўх элементаў азначаецца тыпам падграфа, на дугах якога псеўдапаток ненулявы. Цяпер такіх тыпаў падграфаў пяць:

1) шлях з вяршыні з дэфіцытам у вяршыню з лішкам;
2) генеруючы цыкл разам са шляхам, які пачынаецца ў адной з вяршынь цыкла і заканчваецца ў вяршыні з лішкам;
3) абсарбуючы цыкл разам са шляхам, які пачынаецца ў вяршыні з дэфіцытам і заканчваецца ў адной з вяршынь цыкла;
4) нейтральны цыкл;
5) генеруючы цыкл злучаны шляхам з абсарбуючым цыклам (*біцыкл*).

Цяпер мы азначаем *элементарны абагульнены псеўдапаток* як абагульнены псеўдапаток, для якога

- падграф, утвораны дугамі з дадатным патокам, належыць да аднаго з вышэйазначаных пяці цыпаў;

- лішак усіх вяршынь роўны нулю, акрамя вяршыні, у якую не ўваходзяць дугі (мае дэфіцыт), і вяршыні, з якой не выходзяць дугі (мае лішак).

Тэарэма 5.1 *Кожны абагульнены псеўдапаток g можна прадставіць у выглядзе $g = \sum_{i=1}^{k} g_i$ ($k \le m$), дзе g_i ёсць элементарны абагульнены псеўдапаток.*

Доказ гэтай тэарэмы мы пакідаем чытачу ў якасці вельмі карыснага практыкавання. □

5.2 Абагульненая транспартная задача

Абагульненая транспартная задача заключаецца ў тым каб знайсці дапушчальны абагульнены паток g, які мае мінімальны кошт $c(g)$ сярод ўсіх дапушчальных абагульненых патокаў. Яе фармулёўка як задачы ЛП наступная:

$$\begin{aligned} \sum_{(v,w)\in E} c(v,w)g(v,w) &\to \min \\ -\sum_{(v,w)\in E} g(v,w) &= d(v), \quad v \in V, \\ g(v,w) &= -\gamma(w,v)g(w,v), \quad (v,w) \in E, \\ g(v,w) &\le u(v,w), \quad (v,w) \in E. \end{aligned} \tag{5.7}$$

5.2.1 Прыклады прылажэнняў

Камбінаторная структура абагульненай транспартнай задачы вельмі багатая. Таму яна дазваляе мадэляваць значна больш складаныя практычныя сітуацыіі, у прыватнасці, эканамічныя працэсы. Напрыклад, біцыкл можна выкарыстоўваць ў якасці мадэлі "вытворца-спажывец". Мы ж абмяжуемся разглядам некалькіх мікраэканамічных мадэляў.

Загрузка машын

Маецца r машын, на кожнай з якіх можна вырабляць любы з p прадуктаў. Дапусцім, што машына j можа працаваць толькі на працягу t_j гадзін. Вытворчасць адзінкі прадукта i на машыне j патрабуе τ_{ij} гадзін і каштуе $c_{i,j}$ рублёў. Патрэбна вырабіць α_i адзінак прадукта i. Задача заключаецца ў тым, каб вырабіць патрэбную колькасць прадуктаў з мінімальнымі грашовымі выдаткамі.

Мы сфармулюем гэту задачу як задачу аб абагульненым патоку мінімальнага кошту. Азначым арграф G з мноствам вяршынь

$$V \stackrel{\text{def}}{=} \{1, \ldots, p, 1', \ldots, r'\}$$

і мноствам дуг

$$E \stackrel{\text{def}}{=} \{(i,j') \,:\, i=1,\ldots,p;\; j'=1',\ldots,r'\} \cup \{(j',j') \,:\, j'=1',\ldots,r'\}.$$

Азначым функцыю попыту па правілу:

$$\begin{aligned} d(i) &= -\alpha(i), \quad i=1,\ldots,p, \\ d(j') &= t_j, \quad j'=1',\ldots,r'. \end{aligned}$$

Выйгрыш па дузе (i,j') роўны $\gamma(i,j') \stackrel{\text{def}}{=} \tau_{i,j}$, а кошт гэтай дугі роўны $c(i,j') \stackrel{\text{def}}{=} c_{ij}$. Выйгрыш па дузе (j',j') — $\gamma(j',j') \stackrel{\text{def}}{=} 2$, а яе кошт — $c(j',j') = 0$. Дуга (j',j') уведзена для таго каб "зрасходаваць"час прастою машын і тым самым выканаць умову захавання патока.

5.3 Крытэрыі аптымальнасці

Па аналогіі са звычайнымі патокамі для функцыі цэн $p : V \to \mathbb{R}$ азначым *прыведзены кошт* дугі $(v,w) \in E$ па правілу:

$$c_p^\gamma(v,w) \stackrel{\text{def}}{=} c(v,w) + p(v) - \gamma(v,w)p(w).$$

Лема 5.1 *Для абагульненага псеўдапатока g і функцыі цэн p мае месца наступная роўнасць:*

$$\sum_{(v,w)\in E} c_p^\gamma(v,w)g(v,w) = \sum_{(v,w)\in E} c(v,w)g(v,w) - 2\sum_{v\in V} p(v)e_g^\gamma(v).$$

Доказ. Непасрэдна па азначэнню абагульненага патока маем

$$\begin{aligned} \sum_{(v,w)\in E} c_p^\gamma(v,w)g(v,w) &= \sum_{(v,w)\in E} c(v,w)g(v,w) + \\ &\quad \sum_{v\in V} p(v) \sum_{(w,v)\in E(v,V)} g(v,w) - \\ &\quad \sum_{v\in V} p(v) \sum_{(w,v)\in E(V,v)} \gamma(w,v)g(w,v) \\ &= \sum_{(v,w)\in E} c(v,w)f(v,w) - 2\sum_{v\in V} p(v)e_g^\gamma(v). \end{aligned}$$

□

Тэарэма 5.2 *Абагульнены дапушчальны псеўдапаток g з'яўляецца аптымальным тады і толькі тады, калі існуе функцыя цэн p, такая, што выконваецца наступная ўмова дапаўняючай няжорсткасці:*

$$c_p^\gamma(v,w) \geq 0 \quad \textit{для ўсіх } (v,w) \in E_g\,. \tag{5.8}$$

Доказ. Няхай $\overline{g}$ — адвольны абагульнены паток. Калі $\overline{g}(v,w) > g(v,w)$, то $(v,w) \in E_g$ і па 5.8 атрымліаем, што $c_p^\gamma(v,w) \geq 0$. Аналагічна, калі $g(v,w) > \overline{g}(v,w)$, то $(w,v) \in E_g$ і $c_p^\gamma(w,v) \geq 0$ па (5.8). Таму $c_p^\gamma(v,w) = -c_p^\gamma(w,v) \leq 0$. З улікам гэтага і лемы 5.1 маем

$$\sum_{(v,w)\in E} c(g(v,w) - \overline{g}(v,w)) = \sum_{(v,w)\in E} c_p^\gamma(v,w)(g(v,w) - \overline{g}(v,w)) \geq 0.$$

□

Тэарэма 5.3 *Абагульнены дапушчальны псеўдапаток g з'яўляецца аптымальным тады і толькі тады, калі ў графе G_g няма ні нейтральных цыклаў ні біцыклаў адмоўнага кошту.*

Доказ. Аналагічны доказу тэарэмы 2.1 з выкарыстаннем тэарэмы аб дэкампазіцыі 5.1. □

5.4 Цыклічныя дрэвы

Звязны неарыентаваны граф назавем *цыклічным дрэвам*, калі ён утрымлівае дакладна адзін цыкл. Граф, кожная кампанента звязнасці якога з'яўляецца цыклічным дрэвам, назавем *цыклічным лесам*.

5.4.1 Прадстаўленне цыклічнага лесу

Няхай цыклічны лес складаецца з k цыклічных дрэваў. У i-м цыклічным дрэве bT_i зафіксуем любое рабро e_i (*цыклічнае рабро*), якое належыць адзінаму цыклу ў bT_i. Дрэва $bT_i - e_i$ абазначым праз T_i. Цяпер цыклічны лес можна задаваць спісамі *prec*, *depth*, *next* для задання леса, складзенага з дрэваў $T_1, \ldots, T_k$, плюс адзін дадатковы спіс *cycle_edge*, дзе

$$cycle\_edge[v] \stackrel{\text{def}}{=} \begin{cases} e_i, & v \text{ корань дрэва } T_i\,, \\ \textbf{nil}, & v \text{ не з'яўляецца корнем.} \end{cases}.$$

5.4.2 Азначэнне цэн на вяршынях цыклічнага дрэва

Няхай bT ёсць цыклічнае дрэва ў графе G (без уліку арыентацыі). Так як граф G сіметрычны, то кожнаму рабру $(v,w) \in E(bT)$ адпавядаюць дзве дугі графа G: (v,w) і (w,v). Мы хочам азначыць функцыю цэн p на вяршынях цыклічнага дрэва bT такім чынам, што $c_p^\gamma(v,w) = 0$ для ўсіх $(v,w) \in E(bT)$. Цэны вяршынь будзем спачатку прадстаўляць дзвумя функцыямі $\alpha, \beta : V \to \mathbb{R}$. Гэта азначае, што $p(v) = \alpha(v) + \theta\beta(v)$, дзе параметр θ трэба будзе вызначыць. Няхай r ёсць корань цыклічнага дрэва, а $e = (x,y)$ — яго цыклічнае рабро. Спачатку мы азначаем $\alpha(r) = 0$, $\beta(r) = 1$. Пасля гэтага, выкарыстоўваючы індэксы *next* і ўказальнікі *prec*, рухаемся па

```
compute_prices(parent, next, r, (x, y), p)
{
  α(r) := 0; β(r) := 1;
  for (w := next[r]; w ≠ r; w := next(w)) {
    v := parent(w);
    α(w) := (α(v) + c(v, w))/γ(v, w);
    β(w) := β(v)/γ(v, w);
  }
  θ := (c(x, y) + α(x) − γ(x, y)α(y)) / (γ(x, y)β(y) − β(x));
  p(r) := θ;
  for (v := next[r]; v ≠ r; v := next(v))
      p(v) := α(v) + θβ(v);
}
```

Малюнак 5.1: Працэдура для вылічэння цэн вяршынь цыклічнага дрэва

дугах (v, w) дрэва і вылічваем невядомыя цэны вяршынь w па формуле $p(w) = (c(v, w) + p(v))/\gamma(v, w)$, ці

$$\alpha(w) = (c(v, w) + \alpha(v))/\gamma(v, w), \quad \beta(w) = \beta(v)/\gamma(v, w).$$

У заключэнне, вылічваем θ з умовы

$$c_p^\gamma(x, y) = c(x, y) + \alpha(x) + \theta\beta(x) - \gamma(x, y)(\alpha(y) + \theta\beta(y)) = 0.$$

Адкуль

$$\theta = \frac{c(x, y) + \alpha(x) - \gamma(x, y)\alpha(y)}{\gamma(x, y)\beta(y) - \beta(x)}\,. \tag{5.9}$$

Алгарытм вылічэння цэн прыведзены на мал. 5.4.2.

5.4.3 Азначэнне патокаў на дугах цыклічнага дрэва

Няхай bT ёсць цыклічнае дрэва, $d : V(bT) \to \mathbb{R}$ — функцыя попыту. Пакажам, што на дугах $E(bT)$ можна азначыць адзіны абагульнены псеўпаток f, такі, што $e_f^\gamma = d(v)$ для ўсіх $v \in V(bT)$. Няхай r ёсць корань, а $e = (x, y)$ — цыклічнае рабро bT. Дапусцім, што ўказальнікі *parent*, *next* задаюць дрэва $T \stackrel{\text{def}}{=} bT - e$ Працэдура *compute_flows*, якая будуе дапушчальны абагульнены паток, прадстаўлена на мал. 5.4.3.

У адрозненне ад працэдуры *compute_prices* працэдура *compute_flows* апрацоўвае вяршыні дрэва ў напрамку ад лісцяў да корня. Лішак вяршыні v будзем прадстаўляць як лінейную функцыю $e_f^\gamma(v) \stackrel{\text{def}}{=} \alpha(v) + \theta\beta(v)$ параметра θ, які трэба будзе потым вызначыць. Спачатку для ўсіх вяршынь цыклічнага дрэва $\alpha(v) = d(v)$, а $\beta(v) = 0$ для ўсіх вяршынь, адрозных ад x, y. Так як у далейшым мы азначым паток па дузе (x, y) роўным θ, то азначаем $\beta(x) = 1$, а $\beta(y) = -\gamma(x, y)$.

```
compute_flows(parent, next, r, (x, y), d, f)
{
  α(r) := d(r); β(r) := 0;
  for (w := next(r); w ≠ r; w := next(w)) {
    α(w) := d(w); β(w) := 0;
  }
  β(x) := 1; β(y) := −γ(x, y);
  for (w := next^{−1}(r); w ≠ r; w := next^{−1}(w)) {
    v := parent(w);
    α(v) := α(v) + α(w)/γ(v, w);
    β(v) := β(v) + β(w)/γ(v, w);
  }
  θ := −α(r)/β(r);
  f(x, y) := θ;
  for (w := next(r); w ≠ r; w := next(w)) {
    v := parent(w);
    f(v, w) := −(α(w) + θβ(w))/γ(v, w);
  }
}
```

Малюнак 5.2: Працэдура для вылічэння патокаў на дугах цыклічнага дрэва

Няхай w ёсць нейкі ліст дрэва T, $v = parent(w)$. Адзіны спосаб ліквідаваць дысбаланс велічыні $\alpha(w)+\theta\beta(w)$ у вяршыні w гэта пусціць з вяршыні v па дузе (v, w) паток велічыні $\delta \stackrel{\text{def}}{=} -(\alpha(w) + \theta\beta(w))/\gamma(v, w)$. Што, у сваю чаргу, прывядзе да памяншэння лішку ў вяршыні v на δ. Таму мы замяняем $\alpha(v)$ на $\alpha(v) + \alpha(w)/\gamma(v, w)$, а $\beta(v)$ — на $\beta(v) + \beta(w)/\gamma(v, w)$. Пасля гэтага, мы можам прымяніць апісаны вышэй спосаб і ліквідаваць дысбаланс ва ўсіх вяршынях дрэва акрамя корня r. Выбярэм θ з умовы $\alpha(r) + \theta\beta(r) = 0$. Адкуль $\theta = -\alpha(r)/\beta(r)$. Цяпер і у корні r захоўваецца баланс. Ведаючы θ мы можам паўтарыць усю працэдуру з самага пачатку і азначыць патокі на дугах цыклічнага дрэва.

Мы дакажам карэктнасць нашай працэдуры, калі пакажам, што $\beta(r) \neq 0$. Інакш ураўненне $\alpha(r) + \theta\beta(r) = 0$ не мае рашэння. Успомнім, што дысбаланс велічыні $\alpha(r) + \theta\beta(r)$ утварыўся як вынік таго, што мы паклалі $g(x, y) = \theta$. Гэта, у сваю чаргу, утварыла дэфіцыт велічыні θ у вяршыні x і лішак велічыні $\gamma(x, y)\theta$ у вяршыні y. Няхай $P(r, x)$ ёсць адзіны шлях у дрэве T ад r да x, а $P(y, r)$ — адзіны шлях у дрэве T ад y да r. Адзіны спосаб ліквідаваць дысбаланс у вяршыні x без утварэння дысбаланса ва ўсіх вяршынях акрамя корня — гэта паслаць $\theta/\gamma(P(r, x))$ адзінак патока з корня r уздоўж шляху $P(r, x)$ да вяршыні x. Аналагічна, адзіны спосаб ліквідаваць дысбаланс у вяршыні y без утварэння дысбаланса ва ўсіх вяршынях акрамя корня — гэта паслаць $\gamma(x, y)\theta$ адзінак патока з y уздоўж шляху $P(y, r)$ да r. Усё гэта прывядзе да сумарнага дысбаланса велічыні

$\gamma(x,y)\theta\gamma(P(y,r)) - \theta/\gamma(P(r,x))$ у корні r. Такім чынам,

$$\beta(r) = \gamma(x,y)\gamma(P(y,r)) - \frac{1}{\gamma(P(r,x))}.$$

Таму $\beta(r) = 0$ тады і толькі тады, калі $\gamma(P(r,x))\gamma(x,y)\gamma(P(y,r)) = 1$, што немагчыма, так як $\gamma(P(r,x))\gamma(x,y)\gamma(P(y,r))$ ёсць выйгрыш адзінага цыкла цыклічнага дрэва.

5.5 Абагульнены сеткавы сімплекс-метад

Абагульнены сеткавы сімплекс-метад прыведзены на мал. 5.5. На ўваход алгарытма падаецца дапушчальны абагульнены паток і дапушчальны цыклічны лес, які прадстаўлены функцыямі *parent*, *next*, *depth* і *cycl_edge*. Спачатку алгарытм па цыклічнаму лесу вылічвае функцыю цэн p. Галоўны цыкл алгарытма выклікае працэдуру *check_optimality*, якая правярае ўмову дапаўняючай няжорсткасці. Калі яна выконваецца, то *check_optimality* вяртае значэнне **nil** і метад спыняецца, так як бягучы абагульнены паток аптымальны. У адваротным выпадку, *check_optimality* вяртае дугу (v,w), на якой парушаецца ўмова дапаўняючай няжорсткасці. Потым, спосабам, які будзе апісаны ніжэй, метад знаходзіць дугу (x,y), якая выдаляецца з цыклічнага лесу. У завяршэнне ітэрацыі, алгарытм дабаўляе да цыклічнага лесу дугу (v,w) і пералічвае дугавыя патокі і цэны вяршынь.

5.5.1 Як знайсці дапушчальныя абагульнены паток і цыклічнае дрэва

Існуе вельмі просты штучны шлях пабудаваць дапушчальныя абагульнены паток і цыклічнае дрэва. Азначым дугавыя патокі ў графе G па правілу: $g(v,w) = -u(w,v)$, $g(w,v) = -\gamma(v,w)g(v,w)$. Каб ліквідаваць магчымыя дысбалансы, для кожнай вяршыні $v \in V$ увядзем петлю $e = (v,v)$ і супрацьлеглую $\overline{e} = (v,v)$ з наступнымі параметрамі:

$$u(e) = \infty, \qquad c(e) = M, \qquad \gamma(e) = \begin{cases} \frac{1}{2}, & \text{калі } e_g^\gamma(v) - d(v) \le 0, \\ 2, & \text{калі } e_g^\gamma(v) - d(v) > 0, \end{cases}$$
$$u(\overline{e}) = 0, \qquad c(\overline{e}) = -\gamma(e)c(e), \qquad \gamma(\overline{e}) = \tfrac{1}{\gamma(e)}.$$

дзе M — дастаткова вялікі лік. Для кожнай вяршыні $v \in V$ азначым $g(v,v) \stackrel{\text{def}}{=} \operatorname{abs}(d(v))/(1-\gamma(v,v))$.

Так як кожная штучная пятля мае дастакова вялікі кошт M, то патокі па іх ў аптымальным рашэнні будуць нулявымі. Інакш ў абагульненай патокавай сетцы няма ні воднага дапушчальнага абагульненага патока.

У заключэнене зазначым, што ўсе разам штучныя петлі ўтвараюць дапушчальны цыклічны лес cF, з якога можа пачаць працаваць абагульнены сімплекс-метад.

```
generalized_simplex(G, u, c, γ, d,
                    parent, next, depth, cycl_edge, g, p)
{
  for ( ; r ∈ V; ) if cycl_edge(r) ≠ nil
      compute_prices(parent, next, r, (x, y) := cycl_edge(r), p);
  for ( ; ((v, w) := check_optimality(c, u, g, p)) ≠ nil; ) {
    r1 := find_root(v, parent, next);
    r2 := find_root(w, parent, next);
    for (i := 1; i ≤ n; i := i + 1) d'(i) = 0;
    d'(v) := 1; d'(w) := −γ(v, w);
    compute_flows(parent, next, r1, (x, y) := cycl_edge(r1), d', f);
    if (r1 ≠ r2)
      compute_flows(parent, next, r2, (x, y) := cycl_edge(r2), d', f);
    f(v, w) := 1; f(w, v) := −γ(v, w)f(v, w);
    (x, y) ∈ arg min_{(i,j)∈E: f(i,j)>0} u_g(i,j) / f(i,j);
    g := f + δf;
    Выдаляем рабро (x, y) з базіснага цыклічнага лесу і
    дабаўляем рабро (v, w);
    compute_prices(parent, next, root[i], r1, p);
    if (r1 ≠ r2) {
      compute_prices(parent, next, root[i], r1, p);
    }
  }
}
```

Малюнак 5.3: Абагульнены сеткавы сімплекс-метад

5.5.2 Выбар дугі для вывада з цыклічнага лесу

Няхай мы збіраемся ўводзіць у цыклічнае лес дугу (v, w). Спачатку мы знойдзем адказ на наступнае пытанне:

> на якую велічыню $f(i, j)$ павялічыцца паток па дузе (i, j) цыклічнага леса пры павелічэнні патока па дузе (v, w) на адзінку?

Азначым функцыю попыту b' па правілу:

$$b'(i) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{калі } i = v, \\ -\gamma(v, w), & \text{калі } i = w, \\ 0, & \text{калі } i \in V \setminus \{v, w\}. \end{cases}$$

Няхай f ёсць абагульнены паток з ненулявымі патокамі толькі на дугах цыклічнага лесу, такі, што для кожнай вяршыні $i \in V$ выконваецца балансавая ўмова: $e_f(i) = b'(i)$. Зазначым, што f можна знайсці, выклікаўшы працэдуру *compute_flows* для цыклічных дрэваў, якім належаць вяршыні v і w. Калі v і w належаць аднаму цыклічнаму дрэву, то *compute_flows* трэба выклікаць толькі адзін раз.

Цяпер знойдзем адказ на яшчэ адно пытанне:

на якую максімальную велічыню δ можна павялічыць паток па дузе (v, w), не парушаючы прапускных здольнасцей?

Мы знойдзем δ з наступнай сістэмы няроўнасцяў:

$$g(i,j) + \delta f(i,j) \le u(i,j) \quad \text{для ўсіх } (i,j) \in E',$$

дзе $E' \stackrel{\text{def}}{=} E(bF) \cup (v, w)$. Адкуль

$$\delta = \min_{(i,j)\in E':\, f(i,j)>0} \frac{u_g(i,j)}{f(i,j)}\,. \tag{5.10}$$

Няхай (x, y) ёсць дуга, на якой дасягаецца мінімум у (5.10). Гэтая дуга дуга павінна выдаляцца з цыклічнага леса.

Пасля таго як велічыня δ вызначана, алгарытм павялічвае паток $g(i, j)$ на велічыню $\delta f(i, j)$ для кожнай дугі $(i, j) \in E'$. Калі $\delta = 0$, то ітэрацыя *выраджаная*. У гэтым выпадку абагульнены паток не змяняецца, а змяняецца толькі цыклічны лес (так як $(x, y) \neq (v, w)$).

5.5.3 Геаметрычная інтэрпрэтацыя абагульненага сеткавага сімплекс-метада

Як мы бачылі, сеткавы сімплекс-метад на кожнай ітэрацыі змяняе паток на дугах цыкла адмоўнага кошту ў графе астатніх прапускных здольнасцей. Зараз мы паспрабуем разабрацца, што ж адбываецца на ітэрацыях абагульненага сеткавага сімплекс-метада.

Пры дабаўленні рабра (v, w) да цыклічнага лесу ўтвараецца адзіны біцыкл, скажам (C_1, P, C_2). Магчымы два варыянты: 1) $(v, w) \in E(C_1)$; 2) $(v, w) \in E(P)$. Трэці магчымы выпадак, калі $(v, w) \in E(C_2)$, пернумарацыяй цыклаў C_1 і C_2 зводзіцца да першага.

1) Выбярэм арыентацыю рэбраў цыкла C_1 каб рабро (v, w) праходзілася ў накірунку ад v да w. Калі атрымаем генеруючы цыкл, то арыентуем шлях P у напрамку ад C_1 да C_2, а арыентацыю C_2 выбіраем такой, каб атрымаць абсарбуючы цыкл. Калі ж пасля арыентацыі C_1 з'яуляецца генеруючым цыклам, то выбіраем арыентацыі P і C_2 супрацьлеглымі тым, якія ўказаны вышэй для генерыруючага цыкла.

2) Арыентуем шлях P такім чынам, каб атрымаць дугу (v, w). Калі атрыманы шлях вядзе ад C_1 да C_2, то арыентуем C_1 каб атрымаць генеруючы цыкл, а арыентацыя C_2 выбіраецца з мэтай атрымаць абсарбуючы цыкл. У выпадку, калі пасля арыентацыі шлях P вядзе ад C_2 да C_1, то пасля арыентацыі C_1 павінен быць абсарбуючым, а C_2 — генеруючым.

У абодвух выпадках мы атрымалі дапушчальны арыентаваны біцыкл у графе G_g адмоўнага прыведзенага кошту (усе дугі цыклічнага лесу маюць нулявы кошт, а $c_p^\gamma(v, w) < 0$). Для канкрэтнасці, будзем лічыць, што C_1 — генерыючы цыкл. Адзінку патока, згенерыраванную цыклам C_1, па шляху P можна перавесці ў $\gamma(P)$ адзінак патока на цыкле C_2 і там абсарбаваць

Табліца 5.1: Курсы абмена валют у нейкі дзень

Валюта здадзеная, x	**Валюта атрыманая,** y	**Абменны курс,** r $(y = rx)$	**Ліміт на** x
Долары	Фунты	0.56	1,000
Долары	Ліры	1,241	500
Фунты	Ліры	2,200	160
Ліры	Фунты	0.00045	200,000
Гульдэны	Фунты	3.37	400
Ліры	Йены	0.11	950,000
Йены	Гульдэны	0.014	15,000
Гульдэны	Йены	70.5	500
Гульдэны	Франкі	3.0	1,600
Йены	Франкі	0.042	80,000

іх. У абагульненым сеткавым сімплекс-метадзе біцыкл генерыруе і абсарбіруе столькі многа патока, колькі дазваляюць прапускныя здольнасці дуг біцыкла. Калі пры гэтым па дузе (v, w) будзе працякаць δ адзінак патока, то функцыя мэты паменшыцца на $c_p^\gamma(v, w)\delta$.

5.6 Практыкаванні

1. **Аптымальны абмен валюты**. Кожны дзень у абменным пункце аднаму чалавеку дазволена абмяняць абмежаваную колькасць адной валюты на нейкую колькасць іншай валюты. Гэтыя абмежаванні і абменныя курсы у нейкі канкрэтны дзень прыведзены ў табліцы 1. Дапусцім, што ў вас ёсць 1000 долараў і вы хочаце абмяняць іх на як мага большую колькасць франкаў, выканаўшы нейкую паслядоўнасць абменаў. Сфармулюйце дадзеную задачу як задачу аб абагульненым патоку мінімальнага кошту і вырашыце яе з дапамогай праграмы *NetOpt*.

Литература

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin. Network Flows. Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] Х. Пападимитриу, К. Стайглиц. Комбинаторная оптимизация. Алгоритмы и сложность. – М.: Мир, 1985.

[3] М. Гэри, Д. Джонсон. Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982.

[4] Д. Филлипс, А. Гарсиа-Диас. Методы анализа сетей. – М.: Мир, 1984.

Прадметны ўказальнік