Графы

Н.Н. Писарук
pisaruk@yandex.by

Экономический факультет
Белорусский государственный университет

Минск - 2015

План лекции

1. Графы
   - Деревья
   - Поиск по графу
2. Примеры самых известных задач теории графов
   - Эйлеровы и гамильтоновы циклы
   - Клики, раскраска и укладка графов

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Неориентированные графы

- *Графом* называется пара $G = (V, E)$,
- где V — конечное множество, элементы которого называются *вершинами*,
- а E — это множество *ребер*, каждое из которых представляется парой (v, w) вершин из V.
- Порядок следования вершин не имеет значения: пары (v, w) и (w, v) задают одно и то же ребро.
- Если $e = (v, w) \in E$, то говорят, что вершины v и w *смежны*, и что ребро e *инцидентно* вершинам v и w.
- *Степенью* вершины v, обозначается $\deg(v)$, в графе G называется количество инцидентных ей ребер.
- *Упражнение.* Докажите что сумма степеней всех вершин равна удвоенному числу ребер: $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.
- Это равенство известно как *лемма о рукопожатиях*:
- количество рукопожатий, сделанных всеми гостями на приеме, *четно*.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:

Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:

Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:

Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:

Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

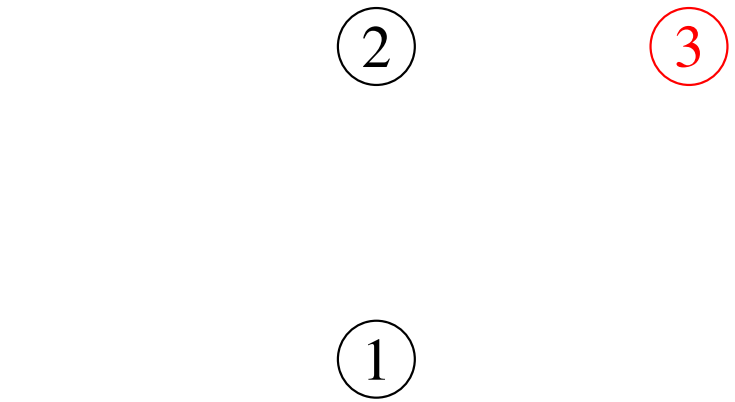
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

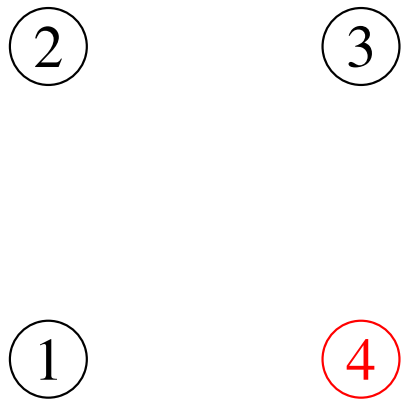
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

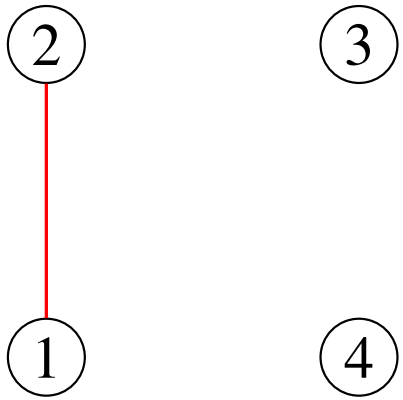
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

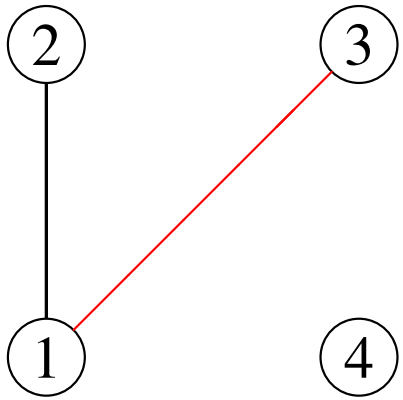
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

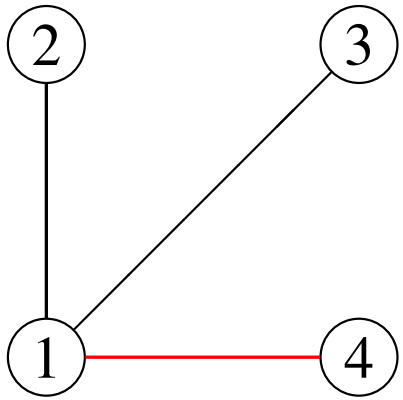
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

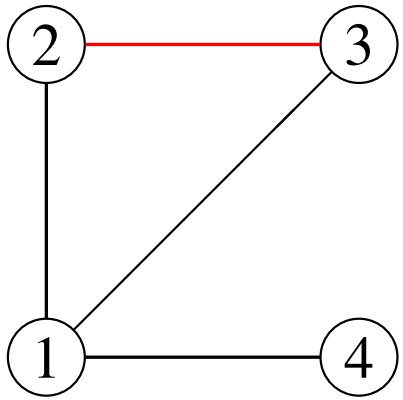
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

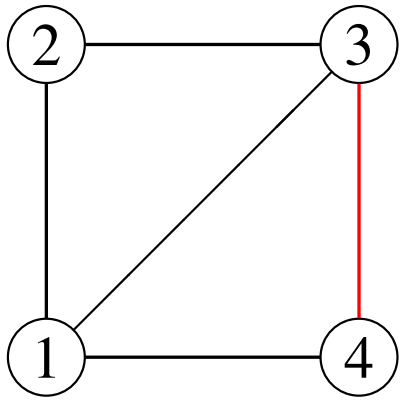
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

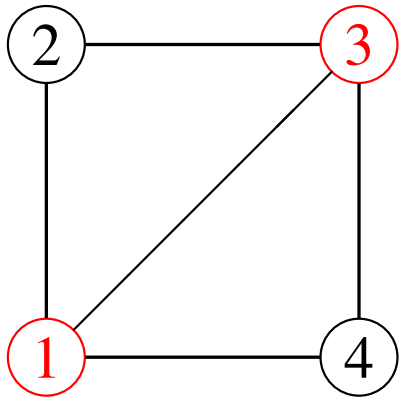
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Рисунок графа

Графы небольшого размера удобно представлять рисунком на плоскости.

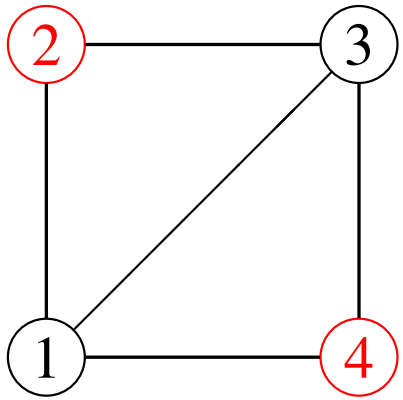
Например, граф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством ребер

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$$

изображается следующим образом:



Здесь степени вершин 1 и 3 равны 3, а степени вершин 2 и 4 равны 2.

Ориентированные графы

- *Ориентированным графом (орграфом)* называется пара $G = (V, E)$,
- где V — конечное множество вершин,
- а E — это множество упорядоченных пар вершин.
- Теперь элементы $e = (v, w)$ множества E называются *дугами*.
- Также говорят, что дуга $e = (v, w)$ выходит из вершины v и входит в вершину w.
- *Степенью исхода* вершины v, обозначается $\mathrm{outdeg}(v)$, называется количество дуг, выходящих из v.
- *Степенью захода* вершины v, обозначается $\mathrm{indeg}(v)$, называется количество дуг, входящих в v.

Ориентированные графы

- *Ориентированным графом* (*орграфом*) называется пара $G = (V, E)$,
- где V — конечное множество вершин,
- а E — это множество упорядоченных пар вершин.
- Теперь элементы $e = (v, w)$ множества E называются *дугами*.
- Также говорят, что дуга $e = (v, w)$ выходит из вершины v и входит в вершину w.
- *Степенью исхода* вершины v, обозначается $\mathrm{outdeg}(v)$, называется количество дуг, выходящих из v.
- *Степенью захода* вершины v, обозначается $\mathrm{indeg}(v)$, называется количество дуг, входящих в v.

Ориентированные графы

- *Ориентированным графом* (*орграфом*) называется пара $G = (V, E)$,
- где V — конечное множество вершин,
- а E — это множество упорядоченных пар вершин.
- Теперь элементы $e = (v, w)$ множества E называются *дугами*.
- Также говорят, что дуга $e = (v, w)$ выходит из вершины v и входит в вершину w.
- *Степенью исхода* вершины v, обозначается $\mathrm{outdeg}(v)$, называется количество дуг, выходящих из v.
- *Степенью захода* вершины v, обозначается $\mathrm{indeg}(v)$, называется количество дуг, входящих в v.

Ориентированные графы

- *Ориентированным графом* (*орграфом*) называется пара $G = (V, E)$,
- где V — конечное множество вершин,
- а E — это множество упорядоченных пар вершин.
- Теперь элементы $e = (v, w)$ множества E называются *дугами*.
- Также говорят, что дуга $e = (v, w)$ выходит из вершины v и входит в вершину w.
- *Степенью исхода* вершины v, обозначается $\mathrm{outdeg}(v)$, называется количество дуг, выходящих из v.
- *Степенью захода* вершины v, обозначается $\mathrm{indeg}(v)$, называется количество дуг, входящих в v.

Ориентированные графы

- *Ориентированным графом* (*орграфом*) называется пара $G = (V, E)$,
- где V — конечное множество вершин,
- а E — это множество упорядоченных пар вершин.
- Теперь элементы $e = (v, w)$ множества E называются *дугами*.
- Также говорят, что дуга $e = (v, w)$ выходит из вершины v и входит в вершину w.
- *Степенью исхода* вершины v, обозначается $\mathrm{outdeg}(v)$, называется количество дуг, выходящих из v.
- *Степенью захода* вершины v, обозначается $\mathrm{indeg}(v)$, называется количество дуг, входящих в v.

Ориентированные графы

- *Ориентированным графом* (*орграфом*) называется пара $G = (V, E)$,
- где V — конечное множество вершин,
- а E — это множество упорядоченных пар вершин.
- Теперь элементы $e = (v, w)$ множества E называются *дугами*.
- Также говорят, что дуга $e = (v, w)$ выходит из вершины v и входит в вершину w.
- *Степенью исхода* вершины v, обозначается $\text{outdeg}(v)$, называется количество дуг, выходящих из v.
- *Степенью захода* вершины v, обозначается $\text{indeg}(v)$, называется количество дуг, входящих в v.

Ориентированные графы

- *Ориентированным графом* (*орграфом*) называется пара $G = (V, E)$,
- где V — конечное множество вершин,
- а E — это множество упорядоченных пар вершин.
- Теперь элементы $e = (v, w)$ множества E называются *дугами*.
- Также говорят, что дуга $e = (v, w)$ выходит из вершины v и входит в вершину w.
- *Степенью исхода* вершины v, обозначается $\mathrm{outdeg}(v)$, называется количество дуг, выходящих из v.
- *Степенью захода* вершины v, обозначается $\mathrm{indeg}(v)$, называется количество дуг, входящих в v.

Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1,2), (1,3), (1,4), (2,3), (3,2), (4,1)\}.$$

Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

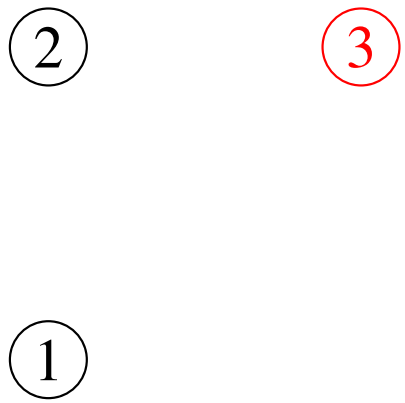


Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

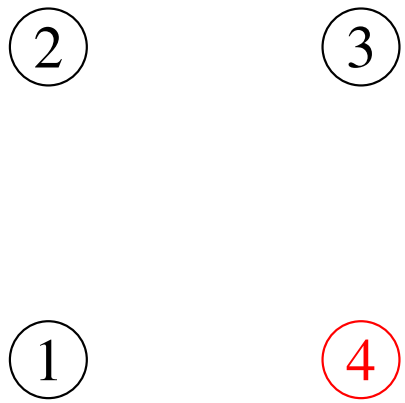


Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

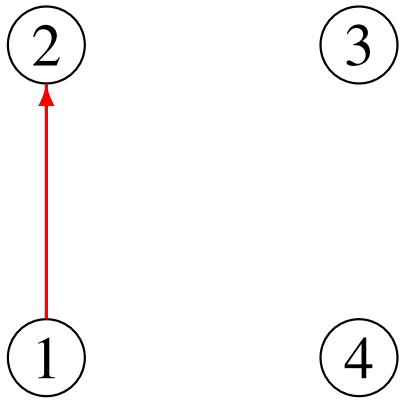


Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

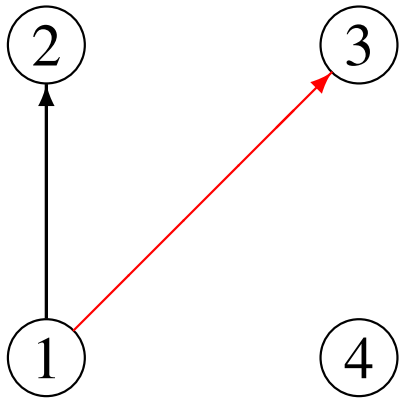


Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

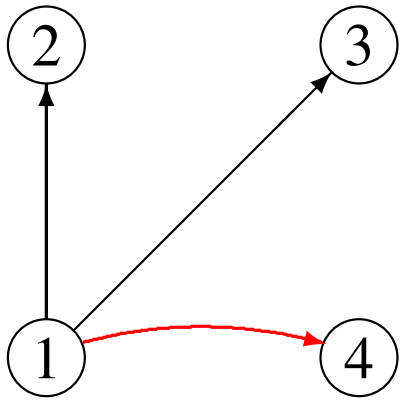


Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$

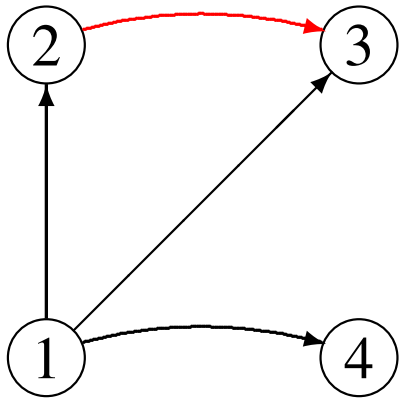


Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1,2), (1,3), (1,4), (2,3), (3,2), (4,1)\}.$$

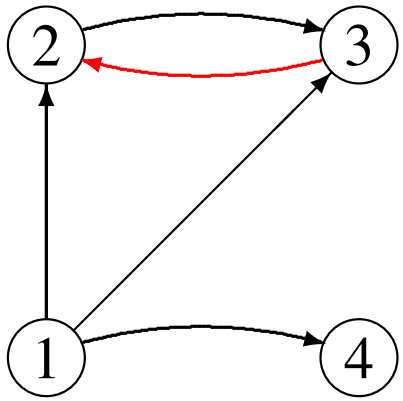


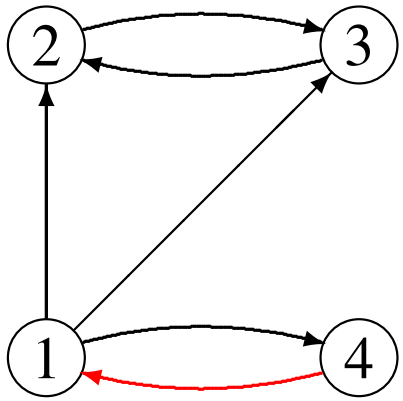
Рисунок орграфа

Нарисуем орграф $G = (V, E)$ с множеством вершин

$$V = \{1, 2, 3, 4\}$$

и множеством дуг

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 2), (4, 1)\}.$$



Мультиграфы

Иногда полезно рассматривать *мультиграфы*,
т. е. графы (орграфы) с кратными (или параллельными) ребрами (дугами).
Пример мультиграфа:

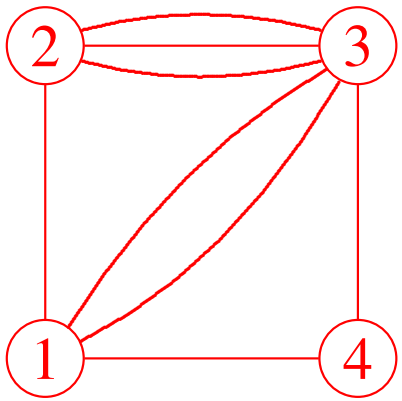
Мультиграфы

Иногда полезно рассматривать *мультиграфы*,
т. е. графы (орграфы) с кратными (или параллельными) ребрами (дугами).
Пример мультиграфа:

Мультиграфы

Иногда полезно рассматривать *мультиграфы*,
т. е. графы (орграфы) с кратными (или параллельными) ребрами (дугами).

Пример мультиграфа:



Подграфы

- Граф $\bar{G} = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$, если
  - $\bar{V} \subseteq V$,
  - и $\bar{E} \subseteq E$.
- Подграфом графа $G = (V, E)$, порожденным множеством вершин $S \subseteq V$, называется подграф $G(S) \stackrel{\text{def}}{=} (S, E(S, S))$,
- где $E(X, Y) \stackrel{\text{def}}{=} \{(v, w) \in E : \ v \in X, \ w \in Y\}$ для $X, Y \subseteq V$.

Подграфы

- Граф $\bar{G} = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$, если
  - $\bar{V} \subseteq V$,
  - и $\bar{E} \subseteq E$.
- Подграфом графа $G = (V, E)$, порожденным множеством вершин $S \subseteq V$, называется подграф $G(S) \stackrel{\text{def}}{=} (S, E(S, S))$,
- где $E(X, Y) \stackrel{\text{def}}{=} \{(v, w) \in E : \ v \in X, \ w \in Y\}$ для $X, Y \subseteq V$.

Подграфы

- Граф $\bar{G} = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$, если
  - $\bar{V} \subseteq V$,
  - и $\bar{E} \subseteq E$.
- Подграфом графа $G = (V, E)$, порожденным множеством вершин $S \subseteq V$, называется подграф $G(S) \stackrel{\text{def}}{=} (S, E(S, S))$,
- где $E(X, Y) \stackrel{\text{def}}{=} \{(v, w) \in E : \ v \in X, \ w \in Y\}$ для $X, Y \subseteq V$.

Подграфы

- Граф $\bar{G} = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$, если
  - $\bar{V} \subseteq V$,
  - и $\bar{E} \subseteq E$.
- Подграфом графа $G = (V, E)$, порожденным множеством вершин $S \subseteq V$, называется подграф $G(S) \stackrel{\text{def}}{=} (S, E(S, S))$,
- где $E(X, Y) \stackrel{\text{def}}{=} \{(v, w) \in E : \ v \in X, \ w \in Y\}$ для $X, Y \subseteq V$.

Подграфы

- Граф $\bar{G} = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$, если
  - $\bar{V} \subseteq V$,
  - и $\bar{E} \subseteq E$.
- Подграфом графа $G = (V, E)$, порожденным множеством вершин $S \subseteq V$, называется подграф $G(S) \stackrel{\text{def}}{=} (S, E(S, S))$,
- где $E(X, Y) \stackrel{\text{def}}{=} \{(v, w) \in E : \ v \in X, \ w \in Y\}$ для $X, Y \subseteq V$.

Пути и циклы

- Последовательность вершин $P = (s = v_0, v_1, \ldots, v_k = t)$ называется *путем* из вершины s в вершину t *длины* k в графе (орграфе) $G = (V, E)$,
- если $(v_{i-1}, v_i) \in E$ для $i = 1, \ldots, k$.
- Путь называется *простым*, если в нем нет повторяющихся вершин.
- Замкнутый (когда $s = t$) путь называют *циклом*.
- *Простой цикл* не имеет повторяющихся вершин.

Пути и циклы

- Последовательность вершин $P = (s = v_0, v_1, \ldots, v_k = t)$ называется *путем* из вершины s в вершину t *длины* k в графе (орграфе) $G = (V, E)$,
- если $(v_{i-1}, v_i) \in E$ для $i = 1, \ldots, k$.
- Путь называется *простым*, если в нем нет повторяющихся вершин.
- Замкнутый (когда $s = t$) путь называют *циклом*.
- *Простой цикл* не имеет повторяющихся вершин.

Пути и циклы

- Последовательность вершин $P = (s = v_0, v_1, \ldots, v_k = t)$ называется *путем* из вершины s в вершину t *длины* k в графе (орграфе) $G = (V, E)$,
- если $(v_{i-1}, v_i) \in E$ для $i = 1, \ldots, k$.
- Путь называется *простым*, если в нем нет повторяющихся вершин.
- Замкнутый (когда $s = t$) путь называют *циклом*.
- *Простой цикл* не имеет повторяющихся вершин.

Пути и циклы

- Последовательность вершин $P = (s = v_0, v_1, \ldots, v_k = t)$ называется *путем* из вершины s в вершину t *длины* k в графе (орграфе) $G = (V, E)$,
- если $(v_{i-1}, v_i) \in E$ для $i = 1, \ldots, k$.
- Путь называется *простым*, если в нем нет повторяющихся вершин.
- Замкнутый (когда $s = t$) путь называют *циклом*.
- *Простой цикл* не имеет повторяющихся вершин.

Пути и циклы

- Последовательность вершин $P = (s = v_0, v_1, \ldots, v_k = t)$ называется *путем* из вершины s в вершину t *длины* k в графе (орграфе) $G = (V, E)$,
- если $(v_{i-1}, v_i) \in E$ для $i = 1, \ldots, k$.
- Путь называется *простым*, если в нем нет повторяющихся вершин.
- Замкнутый (когда $s = t$) путь называют *циклом*.
- *Простой цикл* не имеет повторяющихся вершин.

План лекции

1. Графы
   - Деревья
   - Поиск по графу
2. Примеры самых известных задач теории графов
   - Эйлеровы и гамильтоновы циклы
   - Клики, раскраска и укладка графов

Деревья

- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Деревья

- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Деревья

- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Деревья

- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Деревья

- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Деревья

- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Деревья

- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Деревья

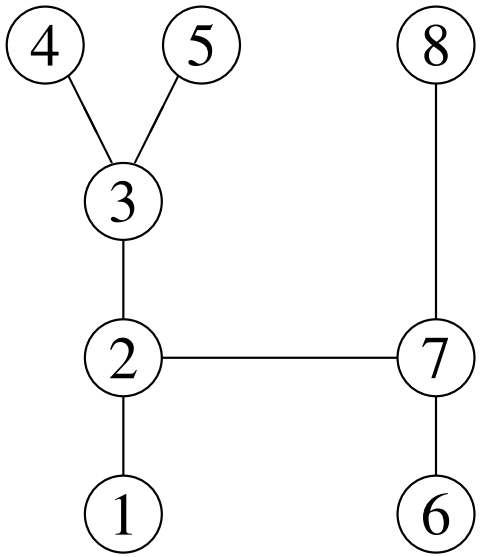
- Граф называется *связным*, если между любыми его двумя вершинами имеется путь.
- *Дерево* — это связный граф без циклов.
- *Лес* — это граф без циклов (или ациклический граф).
- Можно также сказать, что лес — это множество вершинно непересекающихся деревьев.

Теорема 1

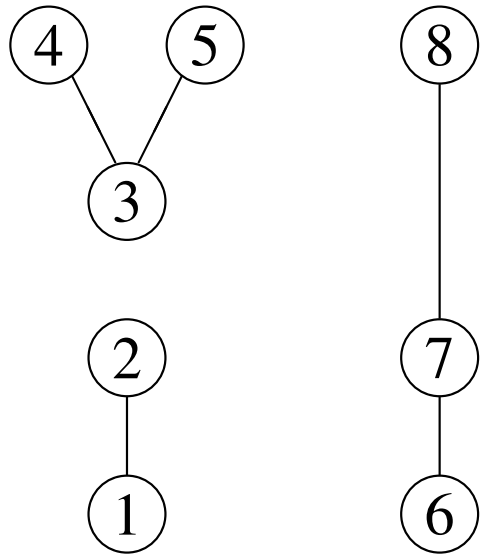
Для графа $G = (V, E)$ следующие условия эквивалентны:

1. *G является деревом;*
2. *G — связный граф с $|V| - 1$ ребрами;*
3. *G не содержит циклов, но при добавлении любого нового ребра к G в нем появится единственный цикл.*

Примеры деревьев



дерево



лес (из 3-х деревьев)

Деревья минимальной стоимости

- *Покрывающим* (или *остовным*) *деревом*) графа $G = (V, E)$ называется
- такой его подграф $T = (V, \bar{E})$ $(\bar{E} \subseteq E)$, который является деревом.
- В задаче *о минимальном остовном дереве*
- в графе $G = (V, E)$, ребрам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$,
- нужно найти остовное дерево $T = (V, \bar{E})$, у которого
- сумма стоимостей его ребер $\sum_{(v,w)\in\bar{E}} c(v, w)$ минимальна.

Деревья минимальной стоимости

- *Покрывающим* (или *остовным*) *деревом*) графа $G = (V, E)$ называется
- такой его подграф $T = (V, \bar{E})$ $(\bar{E} \subseteq E)$, который является деревом.
- В задаче *о минимальном остовном дереве*
- в графе $G = (V, E)$, ребрам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$,
- нужно найти остовное дерево $T = (V, \bar{E})$, у которого
- сумма стоимостей его ребер $\sum_{(v,w)\in\bar{E}} c(v, w)$ минимальна.

Деревья минимальной стоимости

- *Покрывающим* (или *остовным*) *деревом*) графа $G = (V, E)$ называется
- такой его подграф $T = (V, \bar{E})$ $(\bar{E} \subseteq E)$, который является деревом.
- В задаче *о минимальном остовном дереве*
- в графе $G = (V, E)$, ребрам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$,
- нужно найти остовное дерево $T = (V, \bar{E})$, у которого
- сумма стоимостей его ребер $\sum_{(v,w)\in\bar{E}} c(v, w)$ минимальна.

Деревья минимальной стоимости

- *Покрывающим* (или *остовным*) *деревом*) графа $G = (V, E)$ называется
- такой его подграф $T = (V, \bar{E})$ $(\bar{E} \subseteq E)$, который является деревом.
- В задаче *о минимальном остовном дереве*
- в графе $G = (V, E)$, ребрам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$,
- нужно найти остовное дерево $T = (V, \bar{E})$, у которого
- сумма стоимостей его ребер $\sum_{(v,w)\in\bar{E}} c(v, w)$ минимальна.

Деревья минимальной стоимости

- *Покрывающим* (или *остовным*) *деревом*) графа $G = (V, E)$ называется
- такой его подграф $T = (V, \bar{E})$ $(\bar{E} \subseteq E)$, который является деревом.
- В задаче *о минимальном остовном дереве*
- в графе $G = (V, E)$, ребрам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$,
- нужно найти остовное дерево $T = (V, \bar{E})$, у которого
- сумма стоимостей его ребер $\sum_{(v,w)\in\bar{E}} c(v, w)$ минимальна.

Деревья минимальной стоимости

- *Покрывающим* (или *остовным*) *деревом*) графа $G = (V, E)$ называется
- такой его подграф $T = (V, \bar{E})$ $(\bar{E} \subseteq E)$, который является деревом.
- В задаче *о минимальном остовном дереве*
- в графе $G = (V, E)$, ребрам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$,
- нужно найти остовное дерево $T = (V, \bar{E})$, у которого
- сумма стоимостей его ребер $\sum_{(v,w)\in\bar{E}} c(v, w)$ минимальна.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) = \mathbf{nil}$ и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) = \mathbf{nil}$ и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Алгоритм Прима

- *Вход*: граф $G = (V, E)$, функция стоимостей $c : E \to \mathbb{R}$.
- *Выход*: функция $parent : V \to V$, что $\bar{E} = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- *Инициализация*: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;
  - $parent(v) =$ **nil** и $d(v) = \infty$ для всех $v \in V \setminus \{s\}$;
  - $parent(v) = s$, $d(v) = c(s, v)$ для всех $(s, v) \in E(s, V)$.
- Пока $S \neq V$,
  - выбрать $v \in \arg\min\limits_{v \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,
  - и для всех $(v, w) \in E(v, V \setminus S)$, если $d(w) > d(v) + c(v, w)$, положить $parent(w) = v$ и $d(w) = d(v) + c(v, w)$.

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты.
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это и есть задача о минимальном остовном дереве в сети (G, c).

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты.
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это и есть задача о минимальном остовном дереве в сети (G, c).

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты.
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это и есть задача о минимальном остовном дереве в сети (G, c).

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты.
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это и есть задача о минимальном остовном дереве в сети (G, c).

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты.
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это и есть задача о минимальном остовном дереве в сети (G, c).

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты.
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это и есть задача о минимальном остовном дереве в сети (G, c).

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты.
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это и есть задача о минимальном остовном дереве в сети (G, c).

Деревья Штейнера

- Задача построения кратчайшей связующей сети дорог на практике сложнее и не сводится к поиску минимального остовного дерева.
- Проблема в том, что вершины графа должны представлять не только населенные пункты,
- но и перекрестки дорог.
- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы

Деревья Штейнера

- Задача построения кратчайшей связующей сети дорог на практике сложнее и не сводится к поиску минимального остовного дерева.
- Проблема в том, что вершины графа должны представлять не только населенные пункты,
- но и перекрестки дорог.
- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы

Деревья Штейнера

- Задача построения кратчайшей связующей сети дорог на практике сложнее и не сводится к поиску минимального остовного дерева.
- Проблема в том, что вершины графа должны представлять не только населенные пункты,
- но и перекрестки дорог.
- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера*.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера*.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера*.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера*.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера*.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера*.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера*.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,
- а ребра — грунтовые дороги, соединяющие эти населенные пункты и перекрестки .
- Для каждой грутовой дороги $(v, w) \in E$ подсчитана стоимость $c(v, w)$ ее асфальтирования.
- Нужно за минимальную сумму денег
- заасфальтировать некоторые грунтовые дороги, чтобы
- была возможность проехать по асфальтированным дорогам из любого населенного пункта в любой другой населенный пункт.
- Это есть задача поиска в графе G дерева $T = (\bar{V}, \bar{E})$ минимальной стоимости, которое «покрывает» все населенные пункты, т. е. $S \subseteq \bar{V}$.
- Такое дерево называется *деревом Штейнера.*

Ориентированны деревья

- Орграф $G = (V, E)$ называется *ориентированным деревом* (или *ордеревом*),
- если $|E| = |V| - 1$
- и в каждую вершину входит не более одной дуги.
- Для $(v, w) \in E$ вершина v есть *родитель* вершины w (пишем $parent(w) = v$), а w есть потомок вершины v.
- Единственная вершина r в ордереве, в которую не входят дуги, называется *корнем* ($parent(r) = r$).
- Вершины, из которых не выходят дуги, называются *листьями*.

Ориентированны деревья

- Орграф $G = (V, E)$ называется *ориентированным деревом* (или *ордеревом*),
- если $|E| = |V| - 1$
- и в каждую вершину входит не более одной дуги.
- Для $(v, w) \in E$ вершина v есть *родитель* вершины w (пишем $parent(w) = v$), а w есть потомок вершины v.
- Единственная вершина r в ордереве, в которую не входят дуги, называется *корнем* ($parent(r) = r$).
- Вершины, из которых не выходят дуги, называются *листьями*.

Ориентированны деревья

- Орграф $G = (V, E)$ называется *ориентированным деревом* (или *ордеревом*),
- если $|E| = |V| - 1$
- и в каждую вершину входит не более одной дуги.
- Для $(v, w) \in E$ вершина v есть *родитель* вершины w (пишем $parent(w) = v$), а w есть потомок вершины v.
- Единственная вершина r в ордереве, в которую не входят дуги, называется *корнем* ($parent(r) = r$).
- Вершины, из которых не выходят дуги, называются *листьями*.

Ориентированны деревья

- Орграф $G = (V, E)$ называется *ориентированным деревом* (или *ордеревом*),
- если $|E| = |V| - 1$
- и в каждую вершину входит не более одной дуги.
- Для $(v, w) \in E$ вершина v есть *родитель* вершины w (пишем $parent(w) = v$), а w есть потомок вершины v.
- Единственная вершина r в ордереве, в которую не входят дуги, называется *корнем* ($parent(r) = r$).
- Вершины, из которых не выходят дуги, называются *листьями*.

Ориентированны деревья

- Орграф $G = (V, E)$ называется *ориентированным деревом* (или *ордеревом*),
- если $|E| = |V| - 1$
- и в каждую вершину входит не более одной дуги.
- Для $(v, w) \in E$ вершина v есть *родитель* вершины w (пишем $parent(w) = v$), а w есть потомок вершины v.
- Единственная вершина r в ордереве, в которую не входят дуги, называется *корнем* ($parent(r) = r$).
- Вершины, из которых не выходят дуги, называются *листьями*.

Ориентированны деревья

- Орграф $G = (V, E)$ называется *ориентированным деревом* (или *ордеревом*),
- если $|E| = |V| - 1$
- и в каждую вершину входит не более одной дуги.
- Для $(v, w) \in E$ вершина v есть *родитель* вершины w (пишем $parent(w) = v$), а w есть потомок вершины v.
- Единственная вершина r в ордереве, в которую не входят дуги, называется *корнем* ($parent(r) = r$).
- Вершины, из которых не выходят дуги, называются *листьями*.

План лекции

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в начало списка, а pop извлекает элементы из конца списка;
  - *стеком*, если push добавляет элементы в начало списка, и pop тоже извлекает элементы из начала списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в начало списка, а pop извлекает элементы из конца списка;
  - *стеком*, если push добавляет элементы в начало списка, и pop извлекает элементы из начала списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в начало списка, а pop извлекает элементы из конца списка;
  - *стеком*, если push добавляет элементы в начало списка и pop извлекает элементы из начала списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в начало списка, а pop извлекает элементы из конца списка;
  - *стеком*, если push добавляет элементы в начало списка и pop извлекает элементы из начала списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, и pop тоже извлекает элементы из конца списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, а pop извлекает элементы из конца списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, а pop извлекает элементы из конца списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, а pop извлекает элементы из конца списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, а pop извлекает элементы из конца списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, а pop извлекает элементы из конца списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, а pop извлекает элементы из конца списка.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют *списком*.
  - $\{x, y, z\}$ и $\{z, y, x\}$ — одно и то же множество,
  - но (x, y, z) и (z, y, x) — различные списки.
- Две операции над списками:
  - pop(S) извлекает из списка S и возвращает один элемент;
  - push(S, x) добавляет к списку S элемент x.
- Динамически изменяемый с помощью операций pop и push список называется
  - *очередью*, если push добавляет элементы в конец списка, а pop извлекает элементы из начала списка;
  - *стеком*, если push добавляет элементы в конец списка, а pop извлекает элементы из конца списка.

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) = \mathbf{nil}$ для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v \leftarrow pop(Q)$;
    - Для всех $(v, w) \in E(v, V)$, если $parent(w) = \mathbf{nil}$,
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- Алгоритм:
  - *Инициализация:* $Q = (s)$, $parent(s) = s$, $parent(v) = \mathbf{nil}$ для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v \leftarrow pop(Q)$;
    - Для всех $(v, w) \in E(v, V)$, если $parent(w) = nil$,
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска.*

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм:**
  - *Инициализация:* $Q = (s)$, $parent(s) = s$, $parent(v) = \mathbf{nil}$ для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = pop(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) = \mathbf{nil}$,
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска.*

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) = \textbf{nil}$ для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) = \textbf{nil}$,
      полагаем $parent(w) = v$ и помещаем w в $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) = \textbf{nil}$ для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) = \textbf{nil}$, полагаем $parent(w) = v$ и выполняем $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) = \textbf{nil}$ для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) = \textbf{nil}$, **полагаем** $parent(w) = v$ **и выполняем** $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) =$ **nil** для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) =$ **nil**, **полагаем** $parent(w) = v$ **и выполняем** $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) =$ **nil** для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) =$ **nil**,
      полагаем $parent(w) = v$ **и выполняем** $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) =$ **nil** для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) =$ **nil**, **полагаем** $parent(w) = v$ **и выполняем** $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска.*

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) =$ **nil** для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) =$ **nil**, **полагаем** $parent(w) = v$ **и выполняем** $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) =$ **nil** для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) =$ **nil**, **полагаем** $parent(w) = v$ **и выполняем** $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

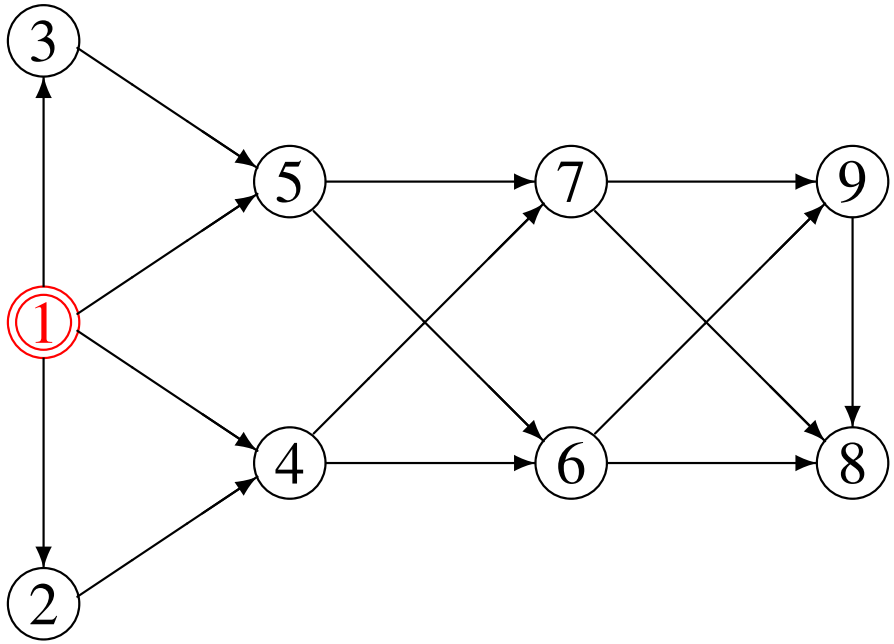
Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- **Алгоритм**:
  - *Инициализация*: $Q = (s)$,
    $parent(s) = s$, $parent(v) =$ **nil** для всех $v \in V \setminus \{s\}$.
  - **Пока** $Q \neq \emptyset$,
    - $v = \text{pop}(Q)$;
    - **Для всех** $(v, w) \in E(v, V)$, **если** $parent(w) =$ **nil**, **полагаем** $parent(w) = v$ **и выполняем** $\text{push}(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают *дерево поиска*.

Поиск по графу называется

- *поиском в ширину*, если Q есть очередь;
- *поиском в глубину*, если Q есть стек;

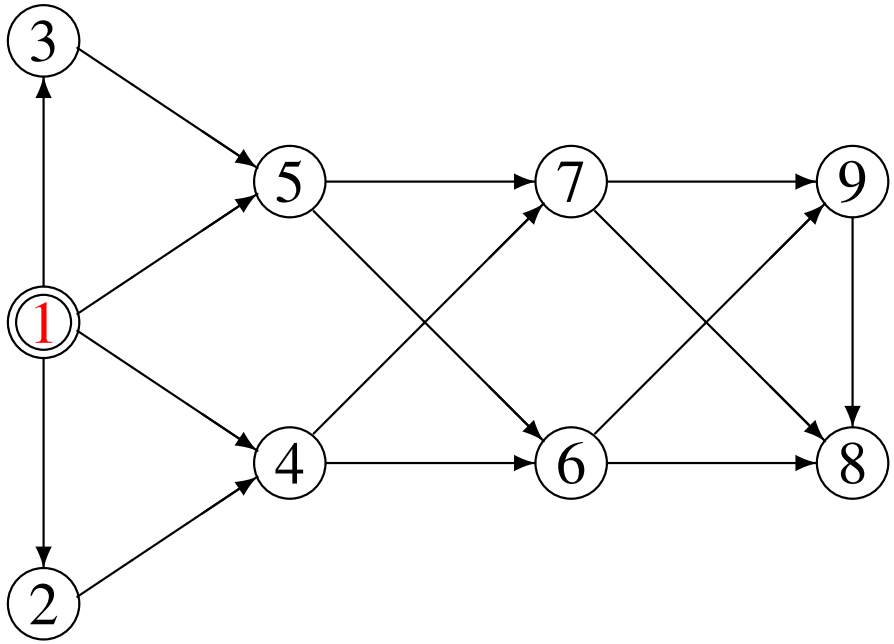
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q

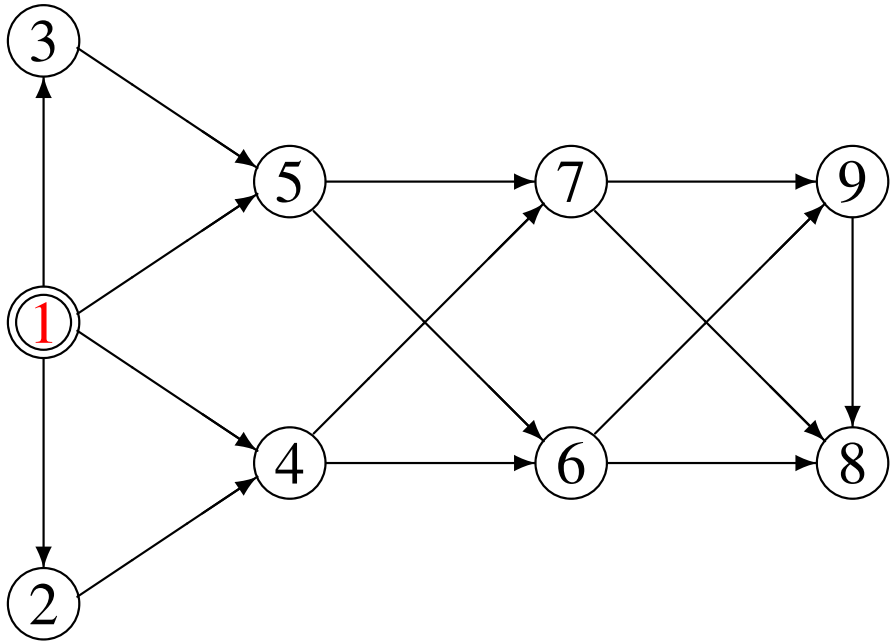
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q

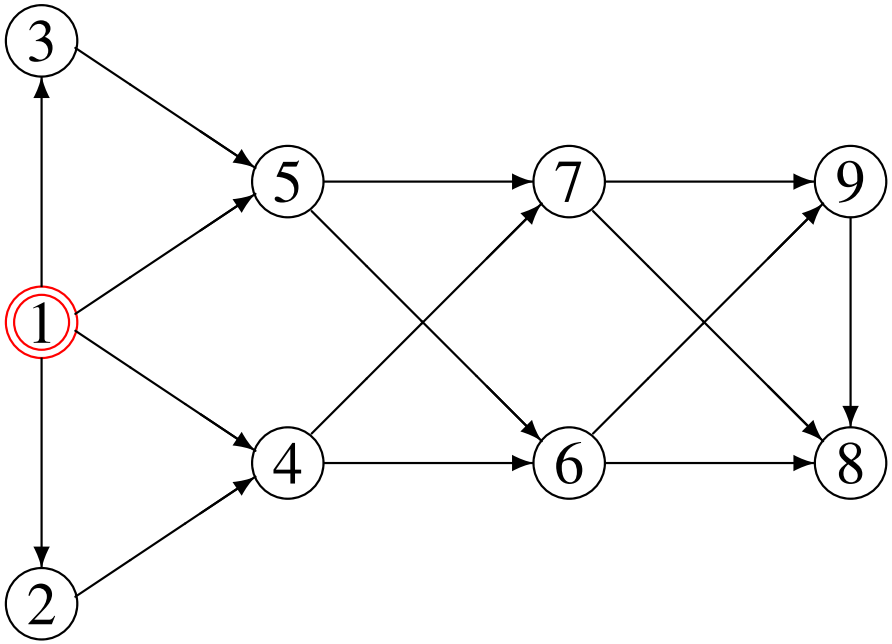
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
	1,

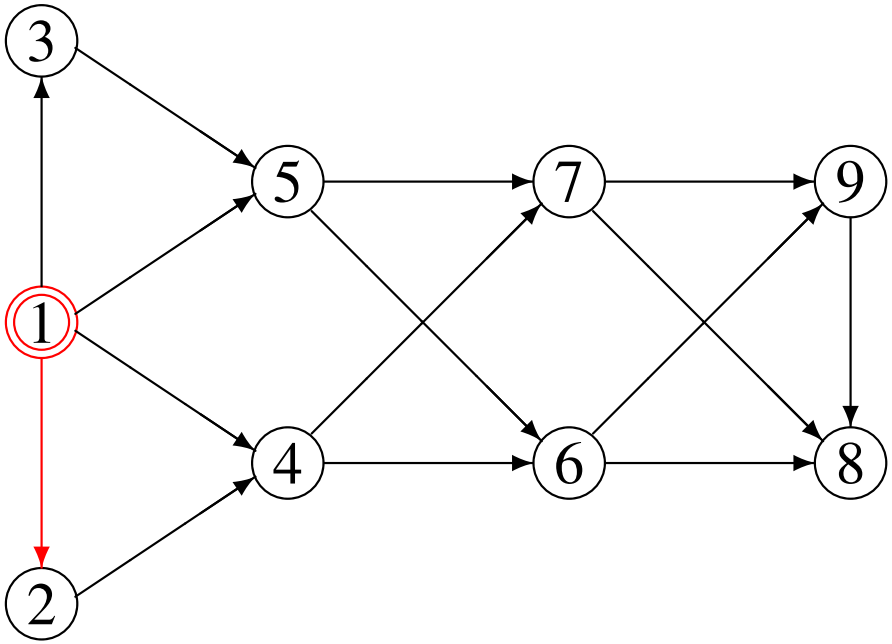
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	

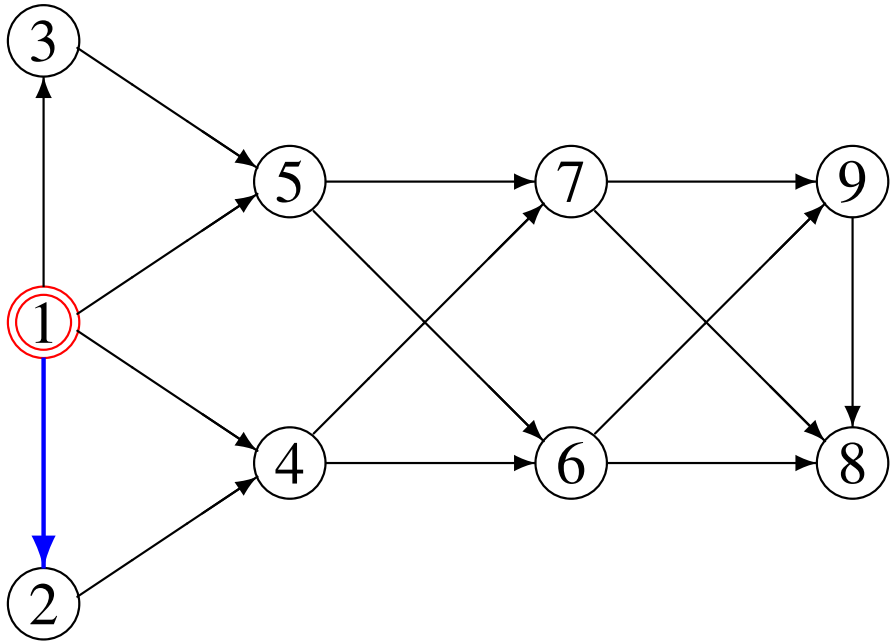
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	

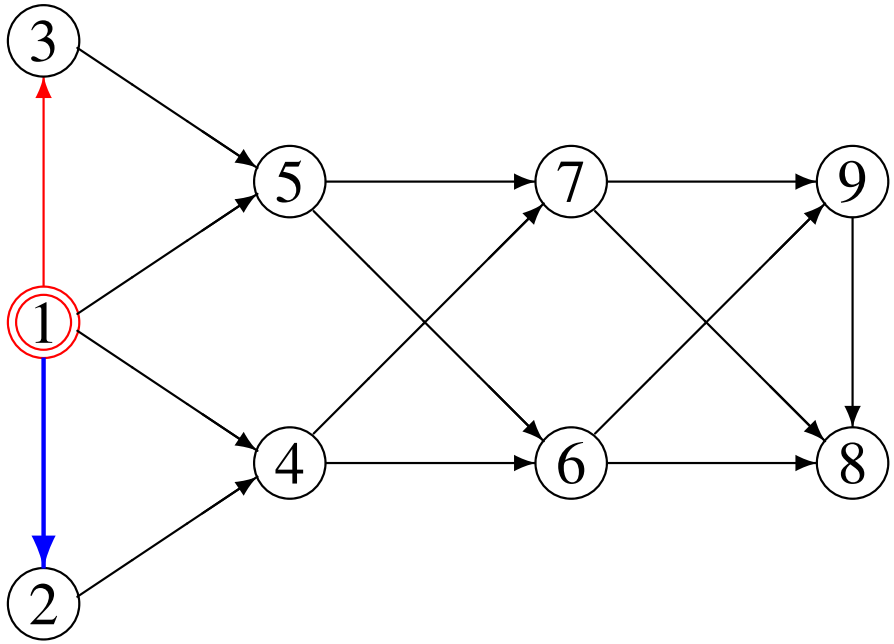
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	2,

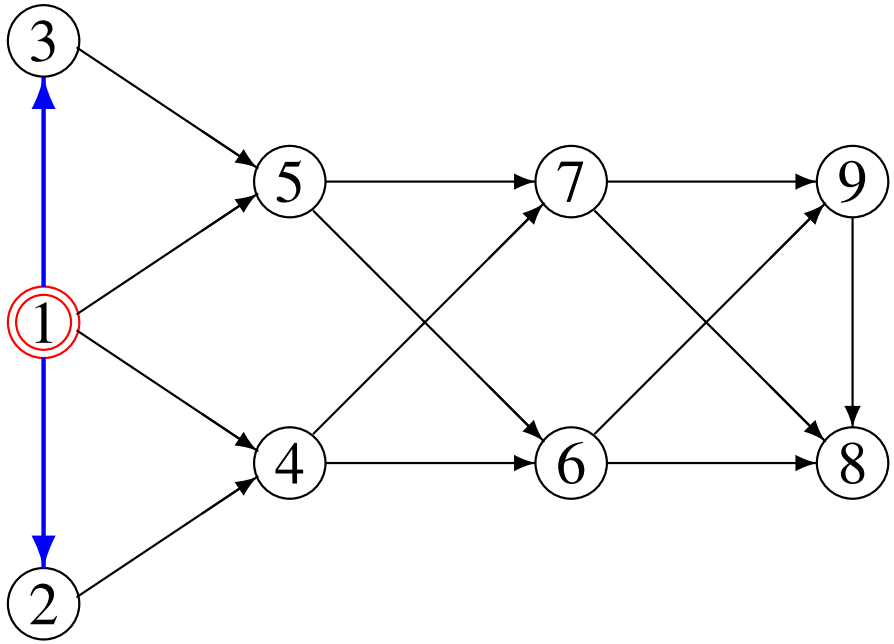
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	2,

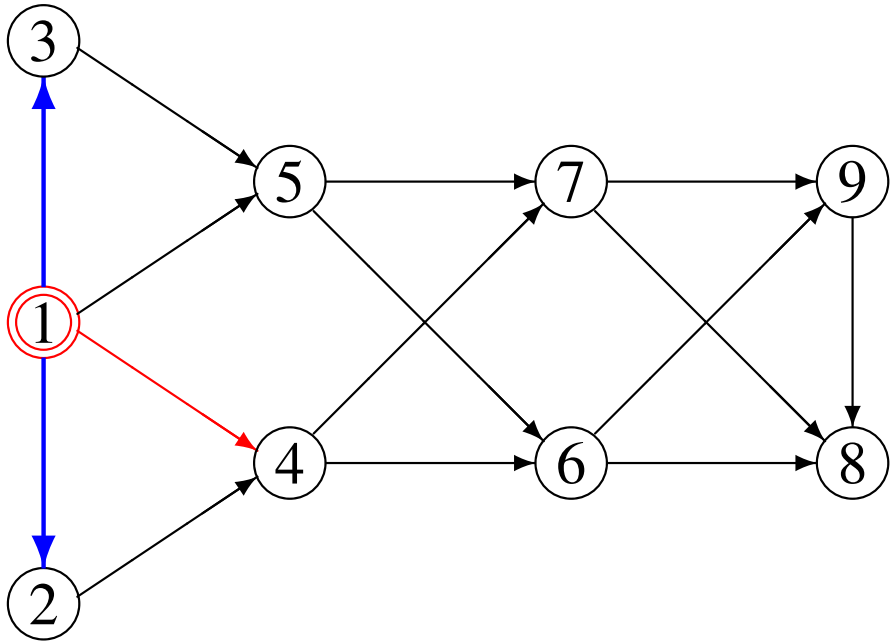
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	2,3,

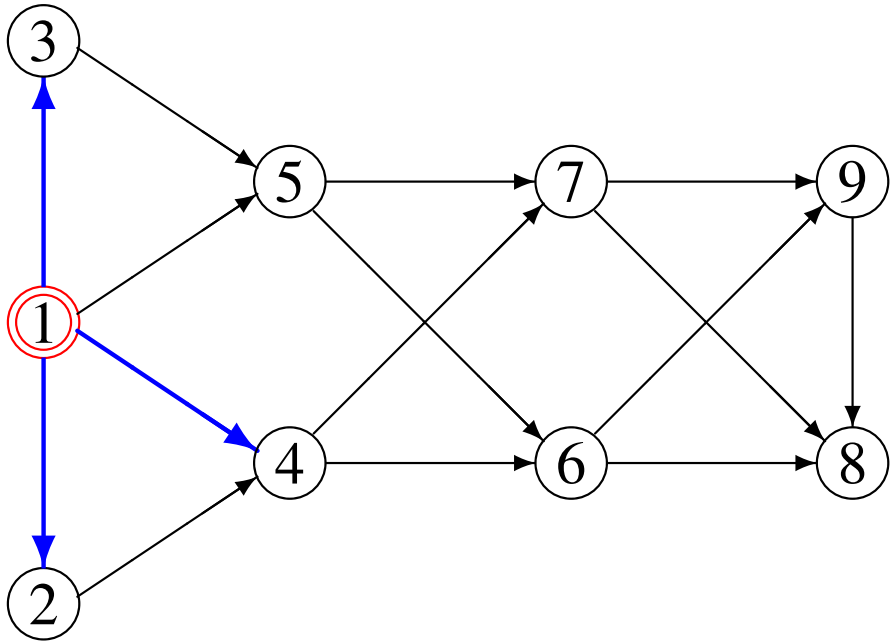
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	**nil**	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	2,3,

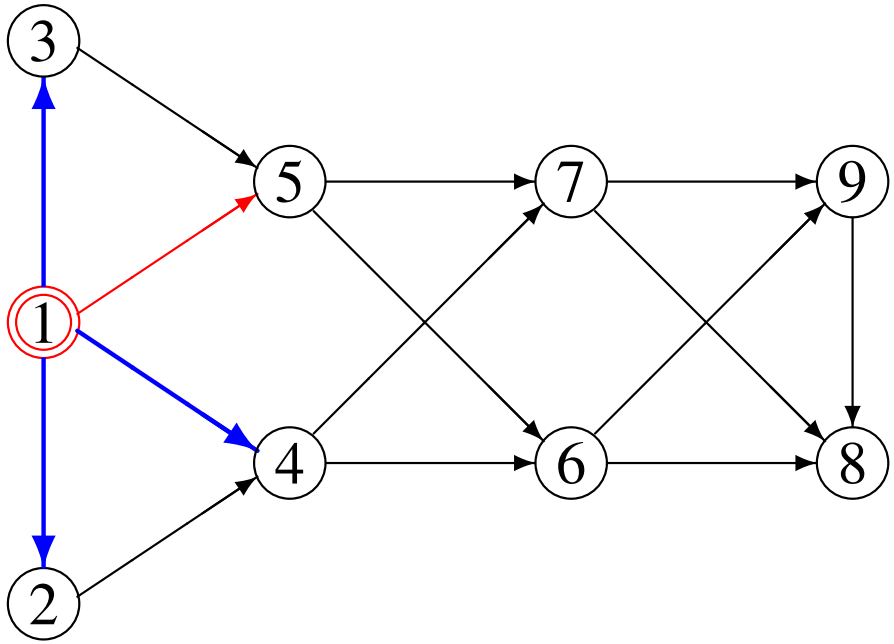
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	2,3,4,

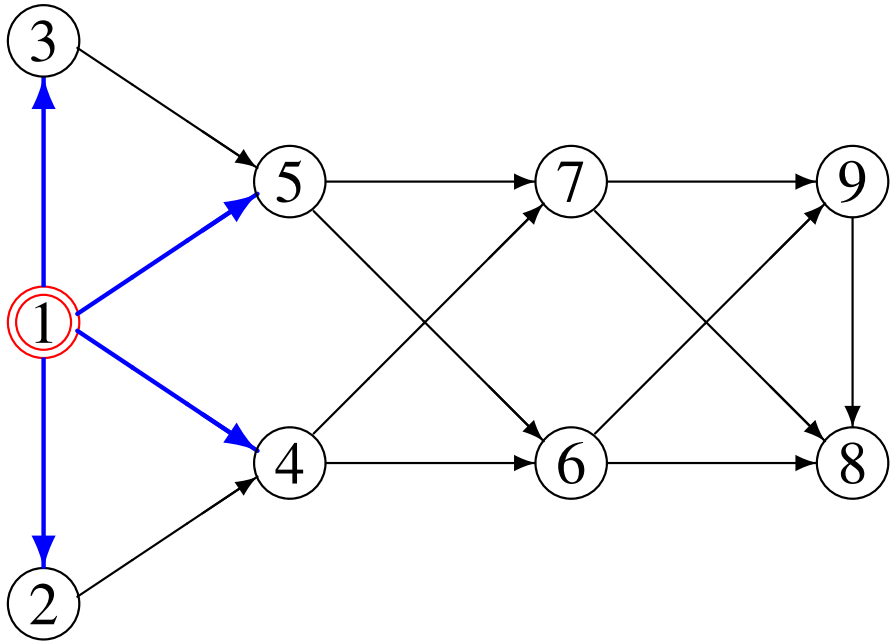
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	**nil**	**nil**	**nil**	**nil**	**nil**

v	Q
1	2,3,4,

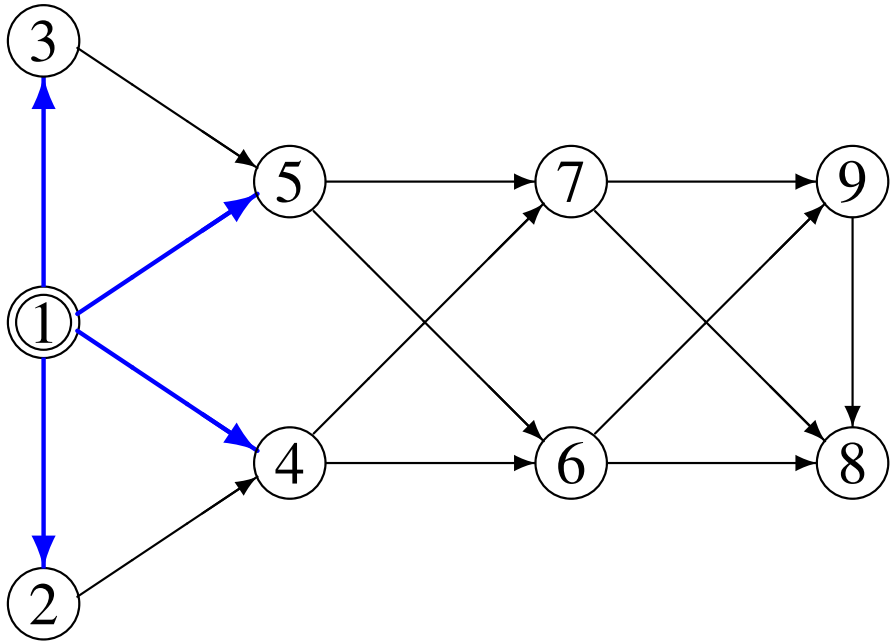
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
1	2,3,4,5,

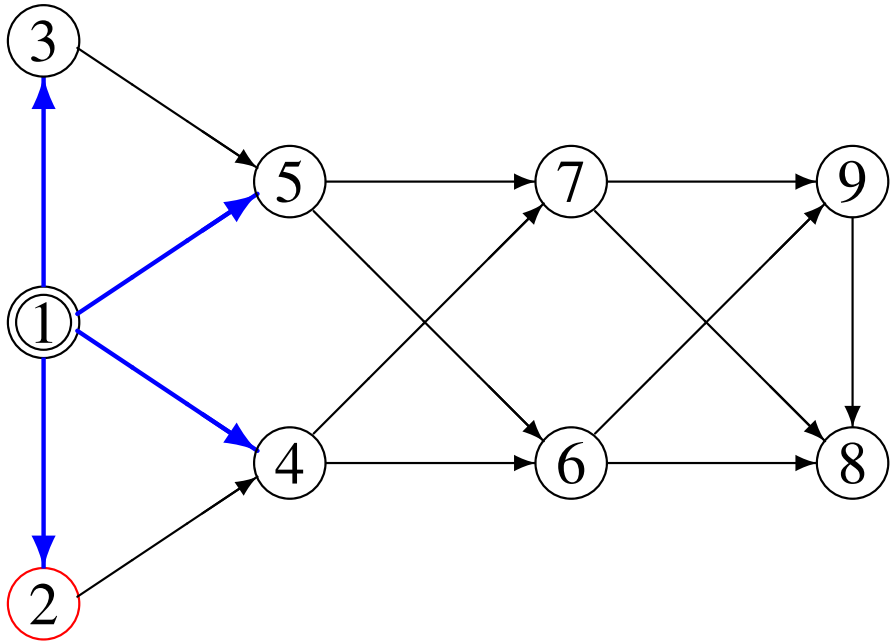
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
	2,3,4,5,

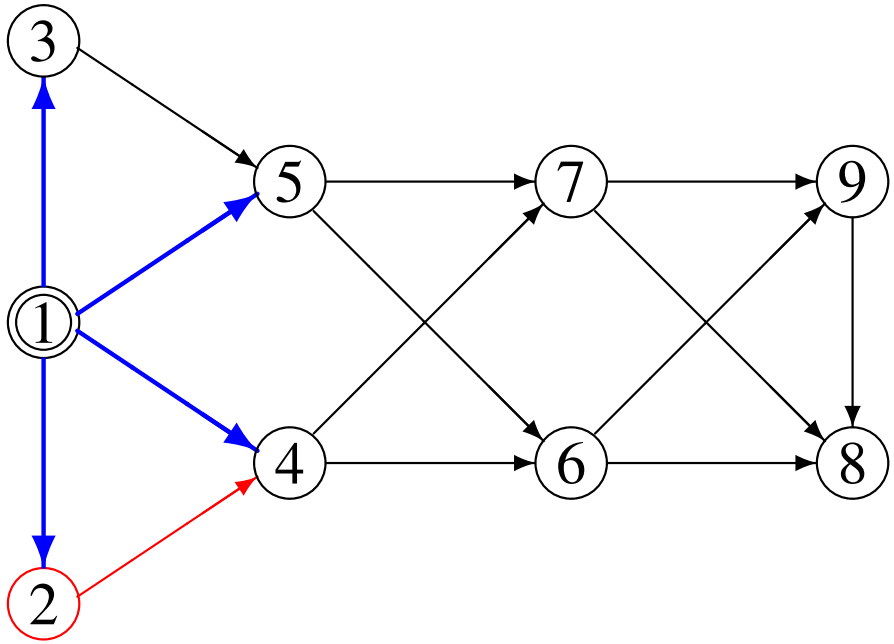
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
2	3,4,5,

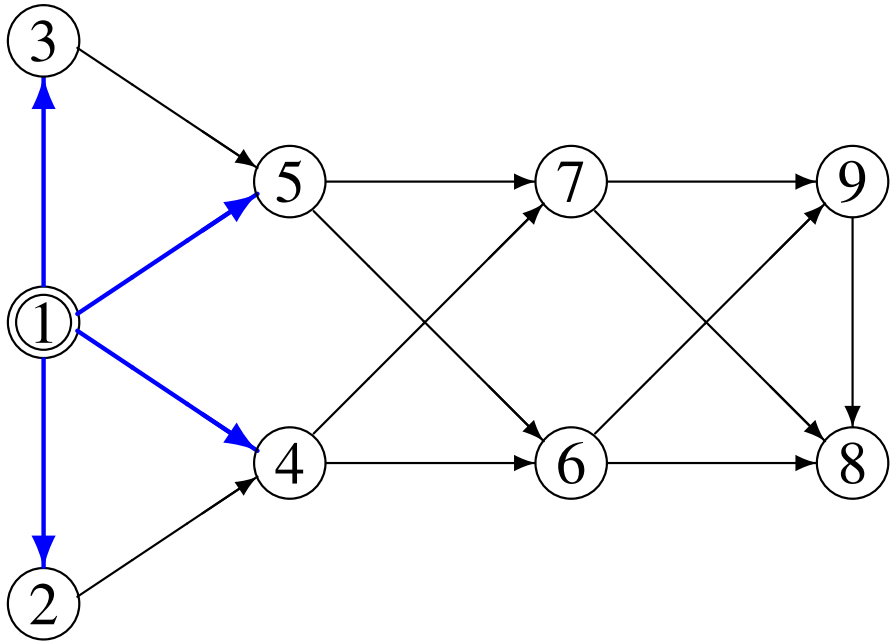
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
2	3,4,5,

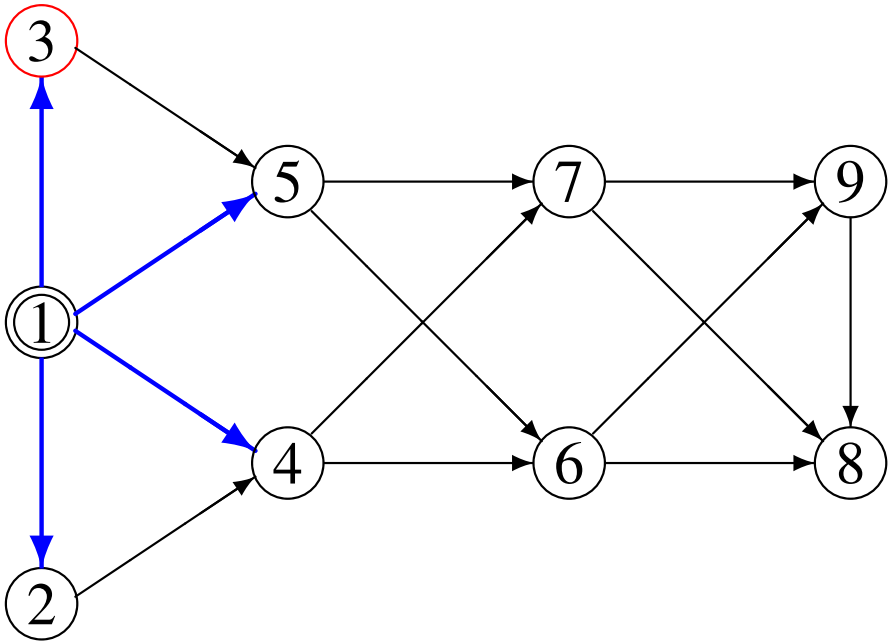
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
	3,4,5,

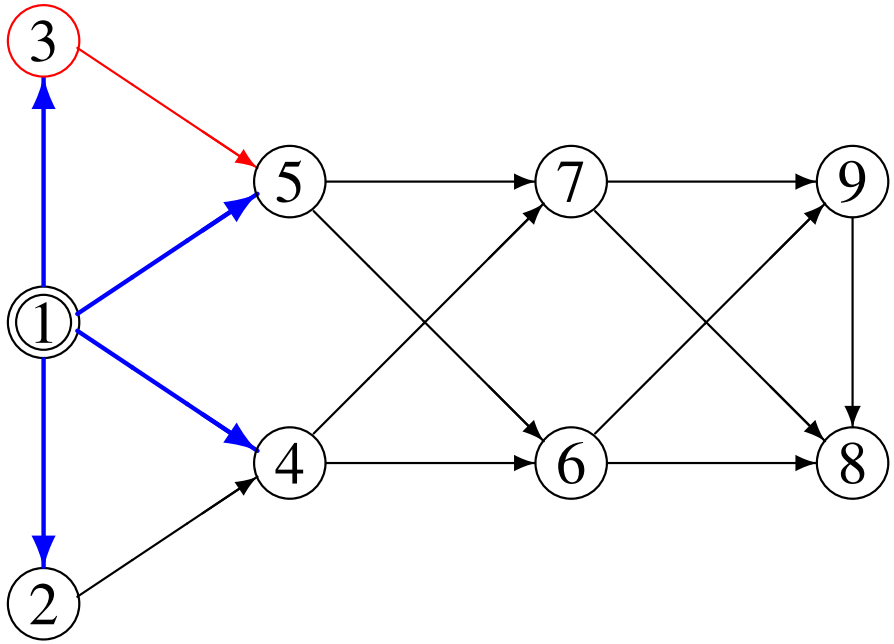
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
3	4,5,

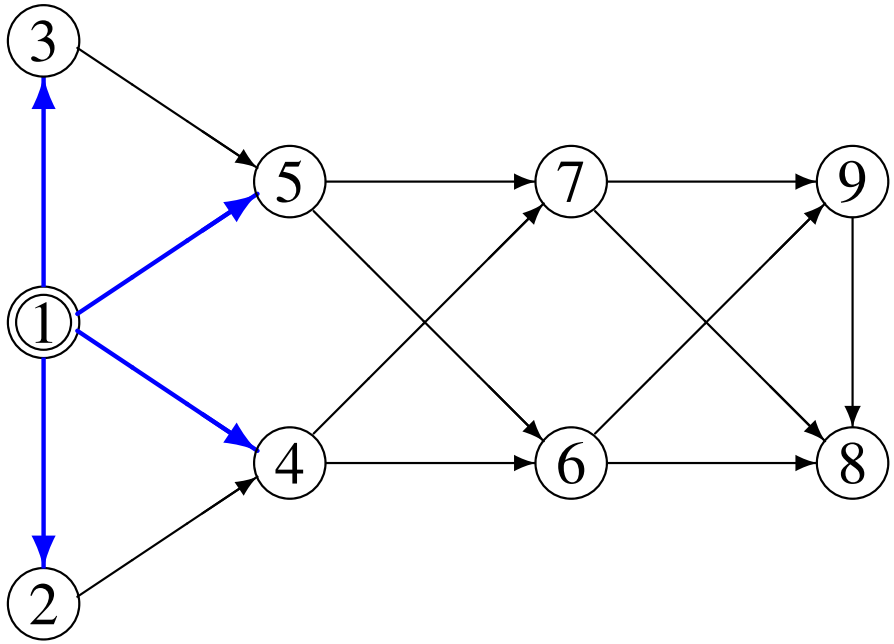
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
3	4,5,

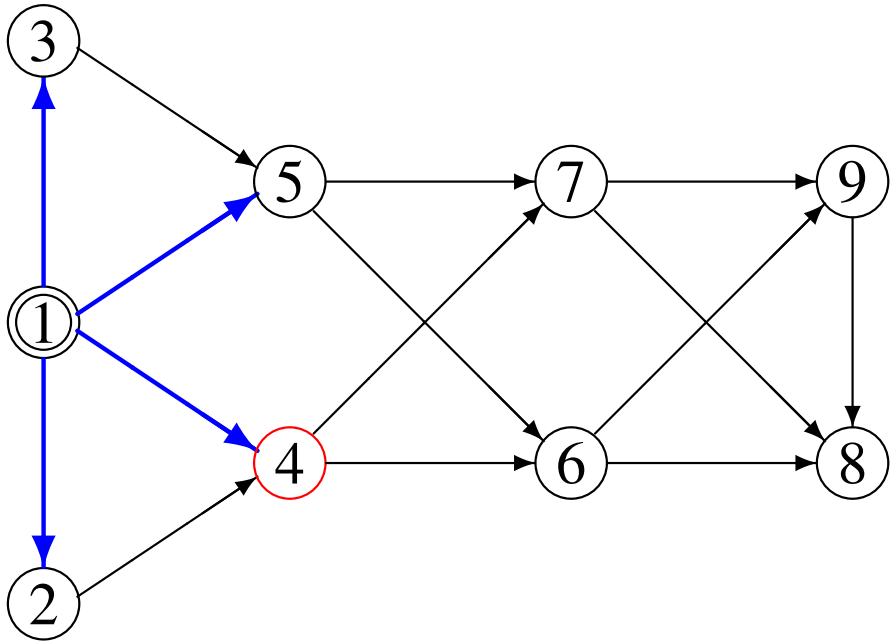
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
	4,5,

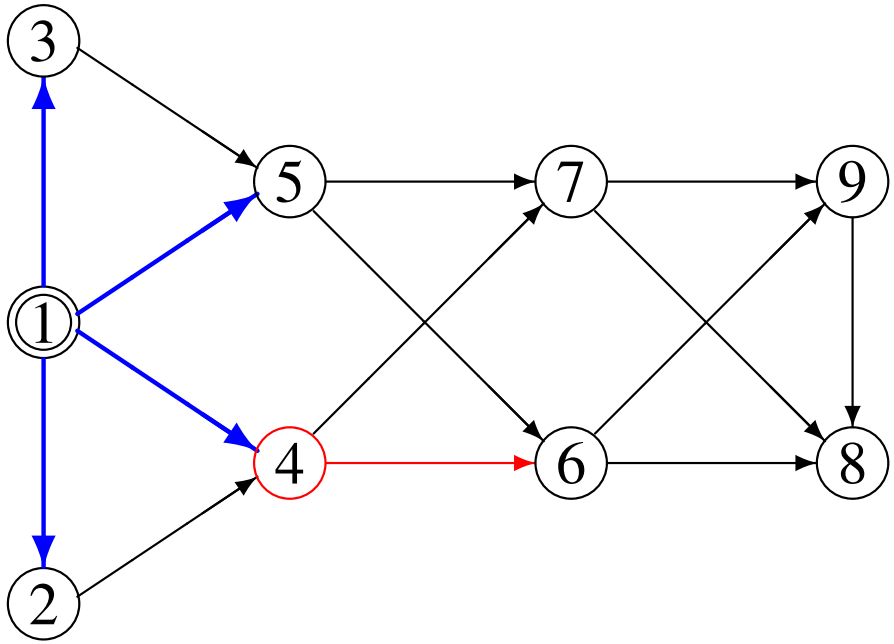
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
4	5,

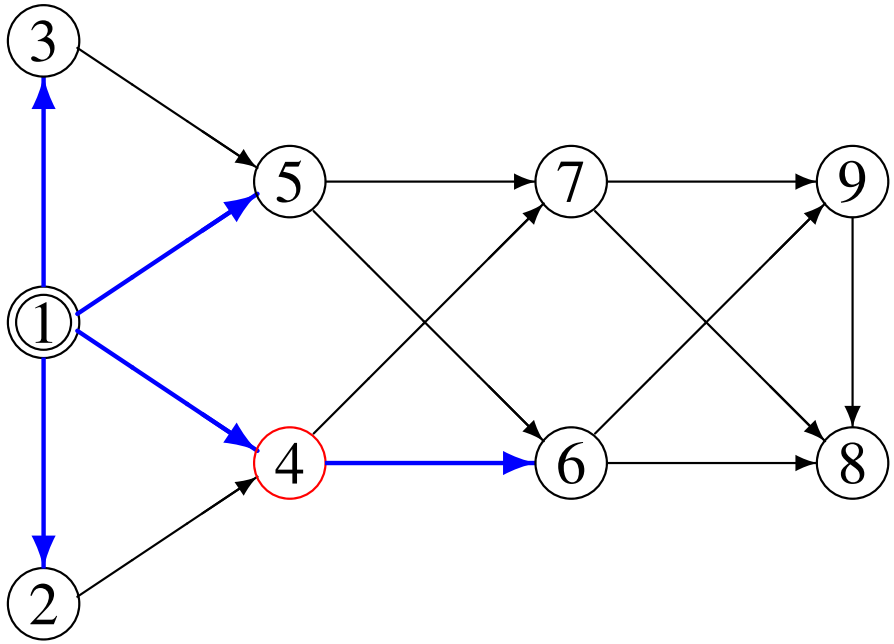
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	**nil**	**nil**	**nil**	**nil**

v	Q
4	5,

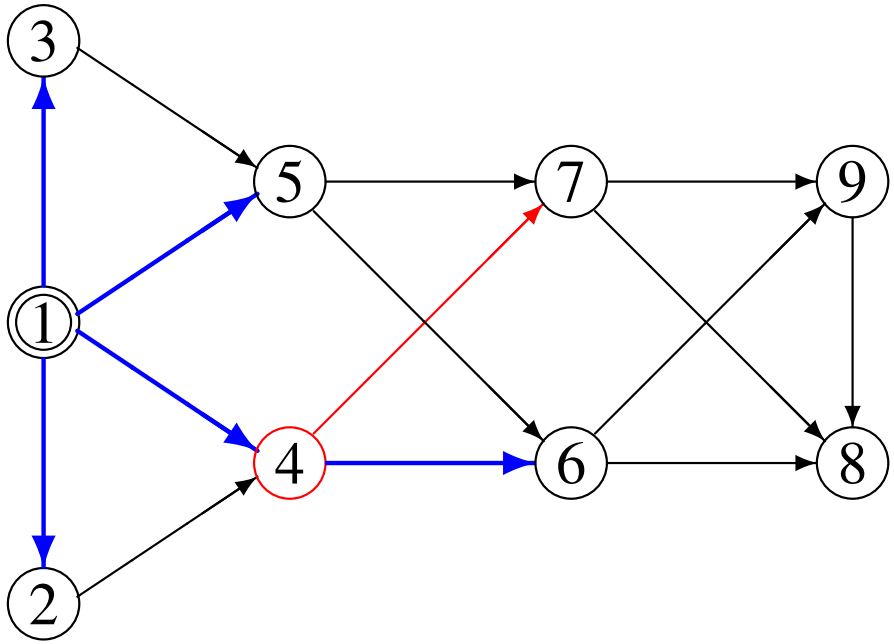
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	**nil**	**nil**	**nil**

v	Q
4	5,6,

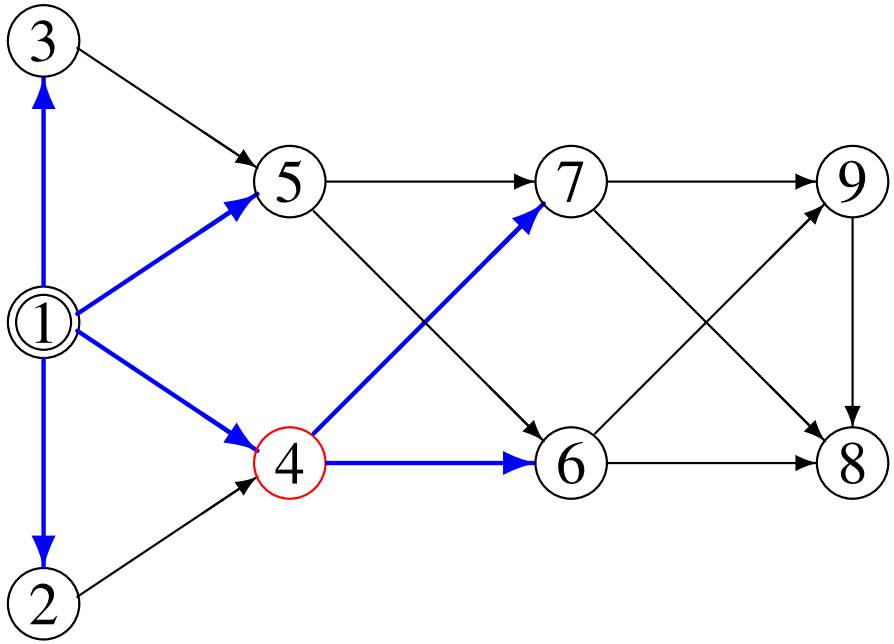
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	**nil**	**nil**	**nil**

v	Q
4	5,6,

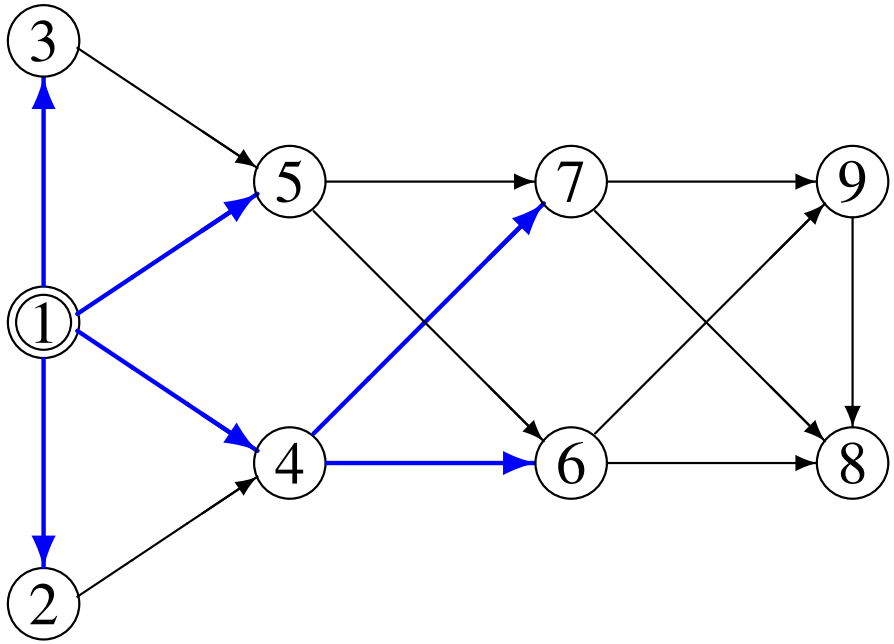
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
4	5,6,7,

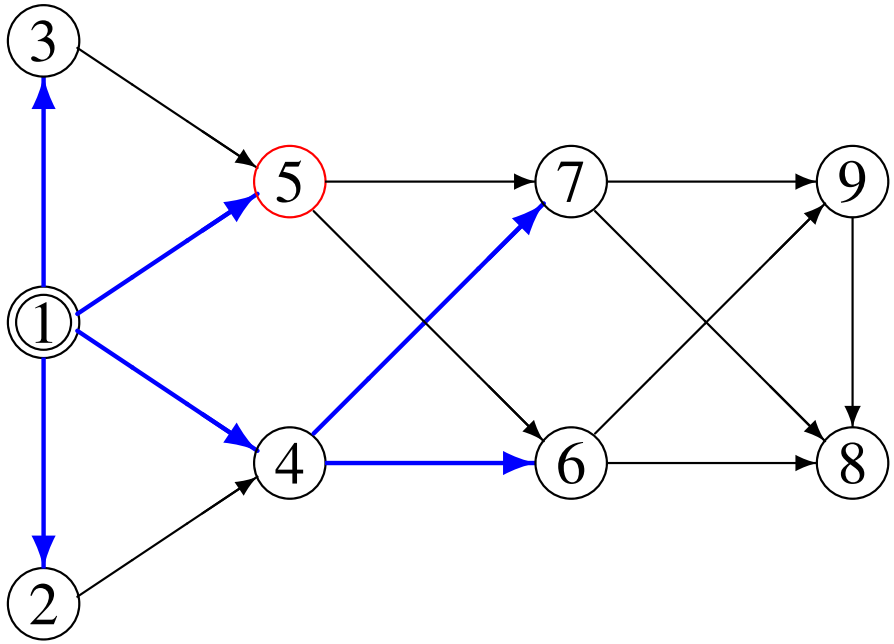
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
	5,6,7,

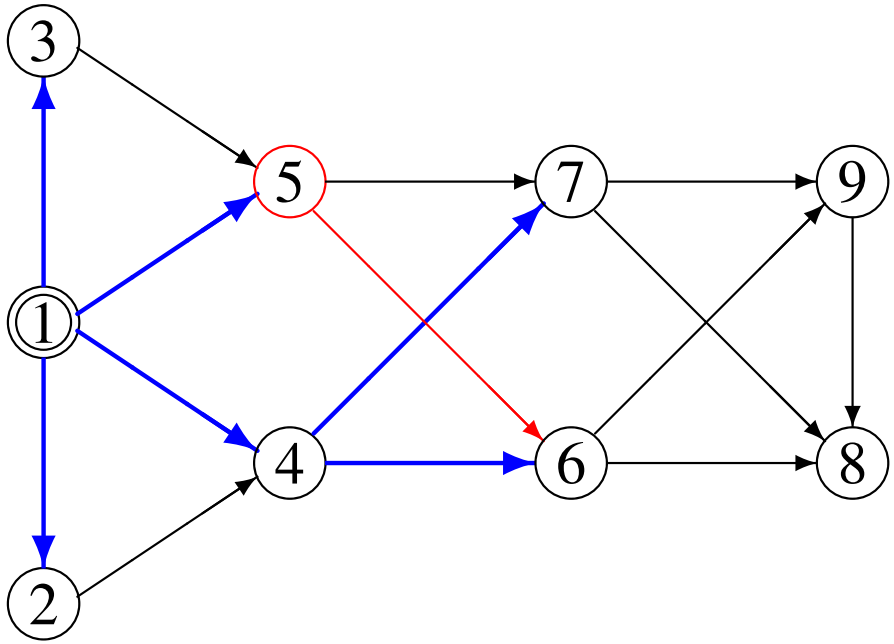
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
5	6,7,

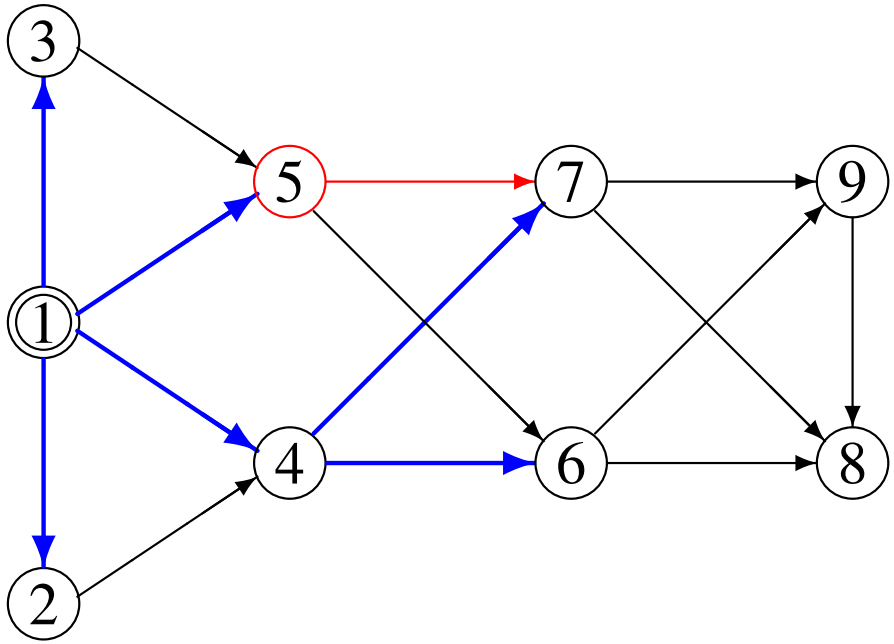
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
5	6,7,

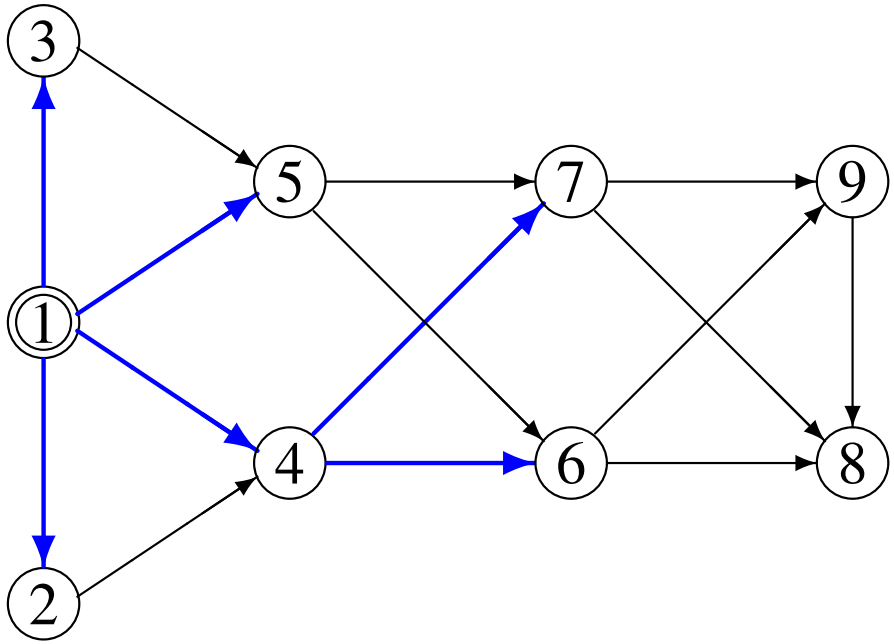
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
5	6,7,

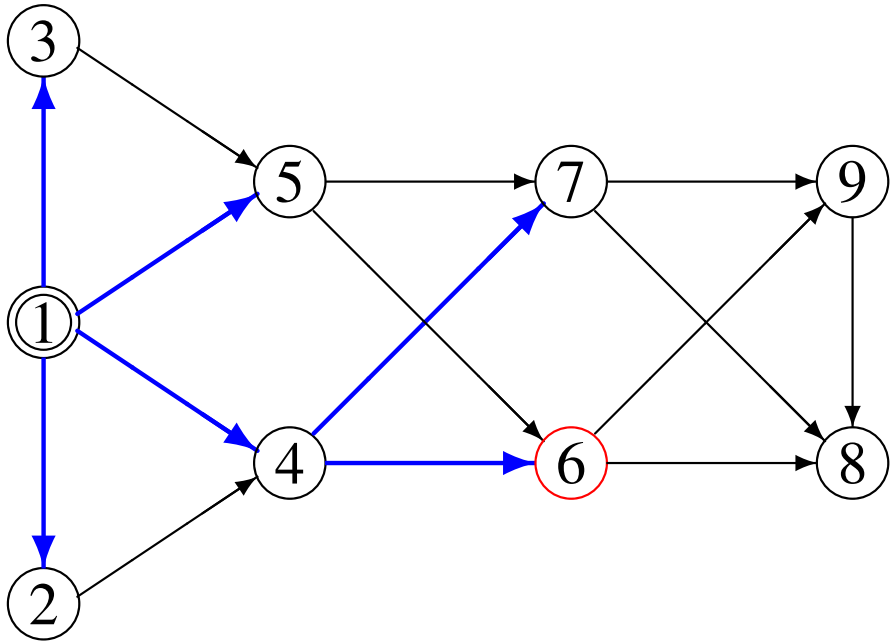
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
	6,7,

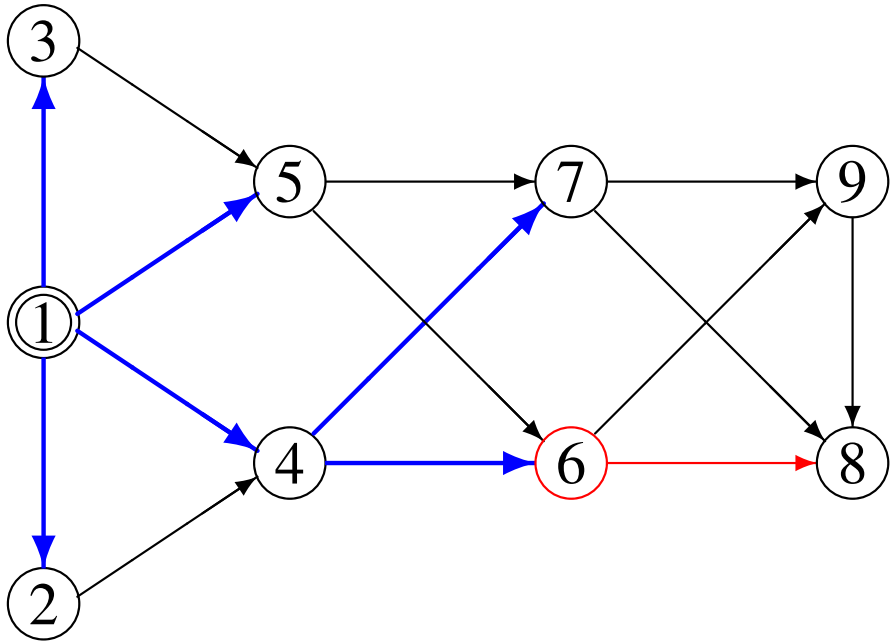
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
6	7,

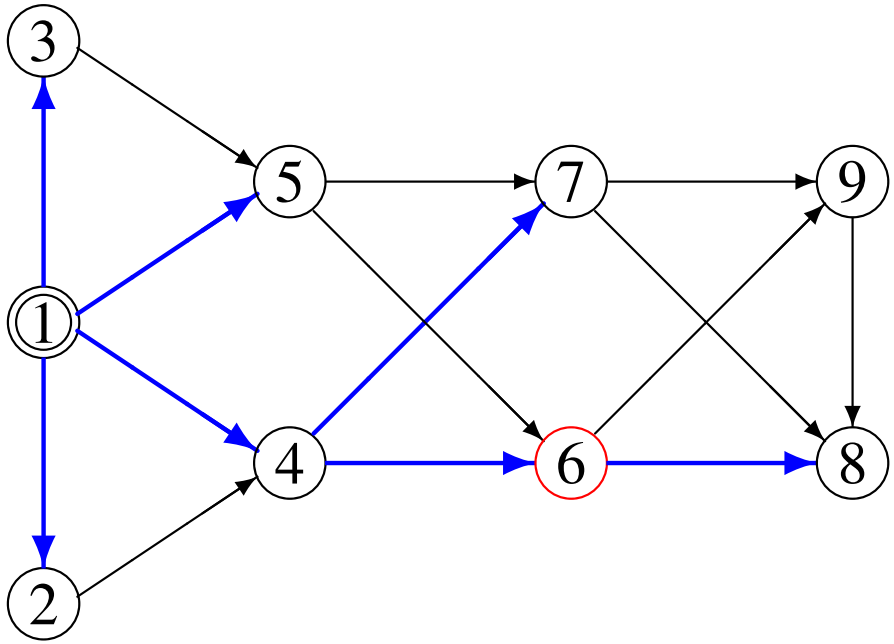
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	**nil**	**nil**

v	Q
6	7,

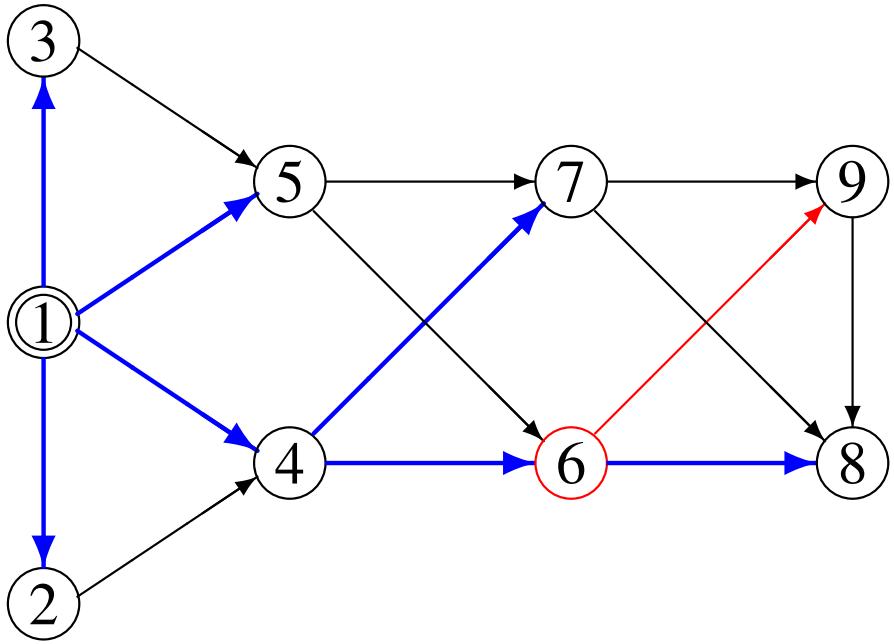
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	**nil**

v	Q
6	7,8,

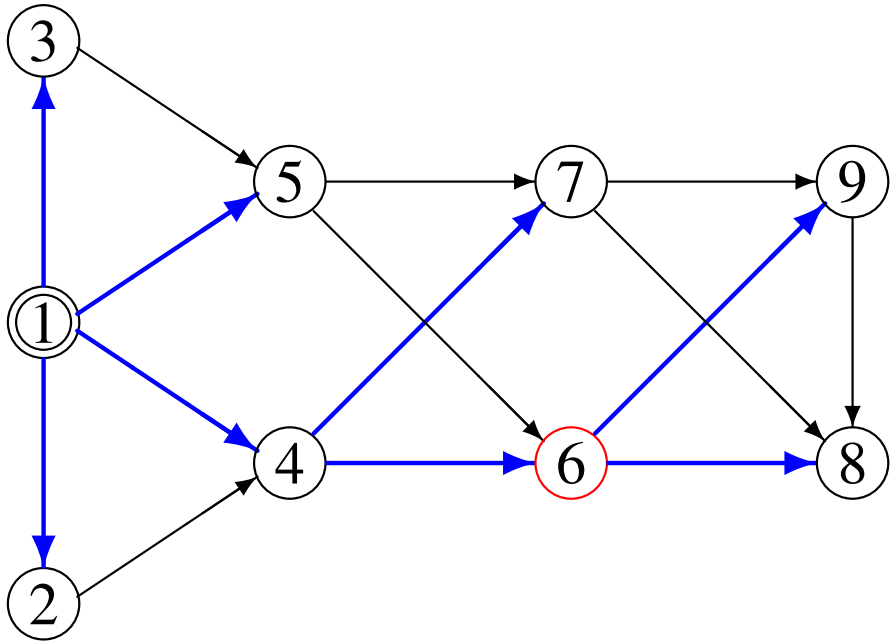
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	**nil**

v	Q
6	7,8,

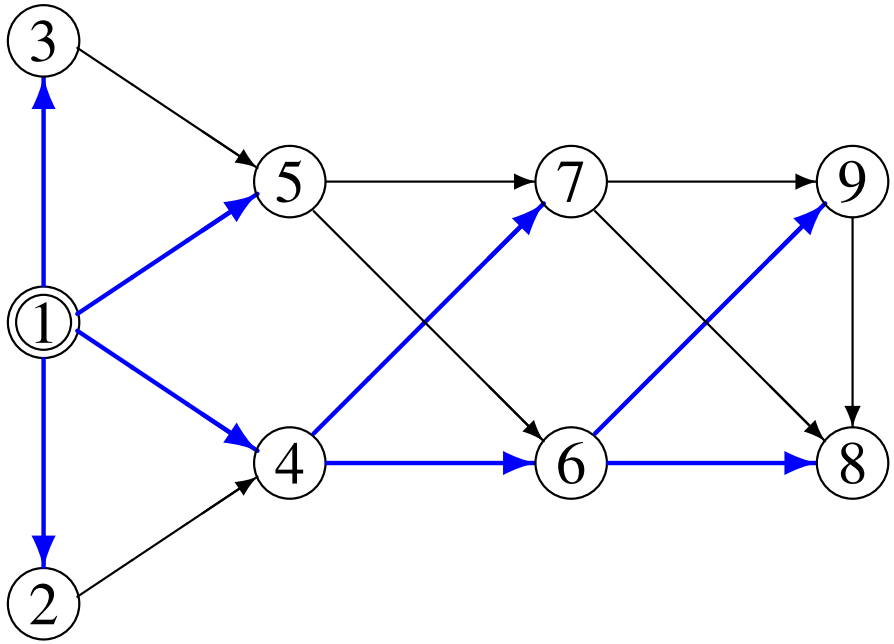
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
6	7,8,9,

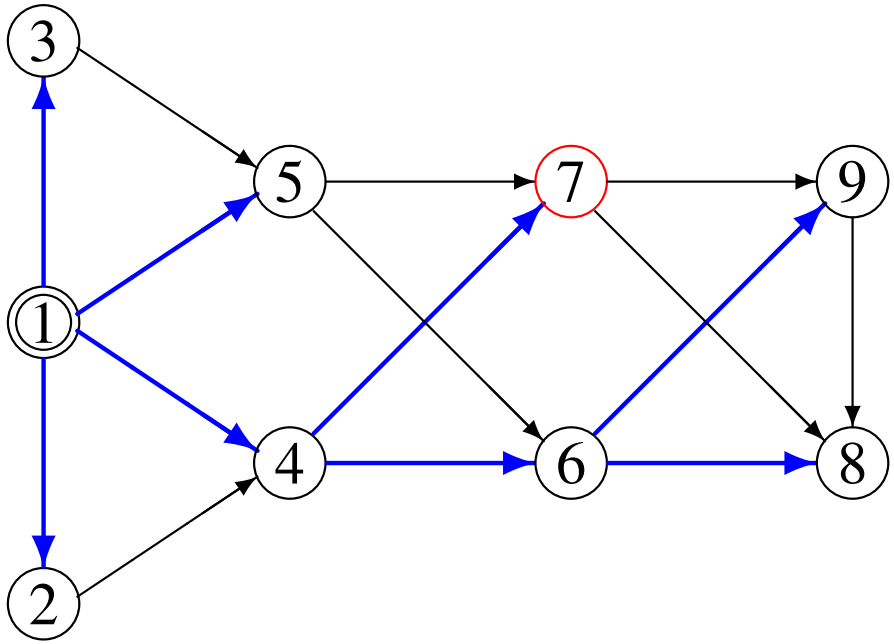
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
	7,8,9,

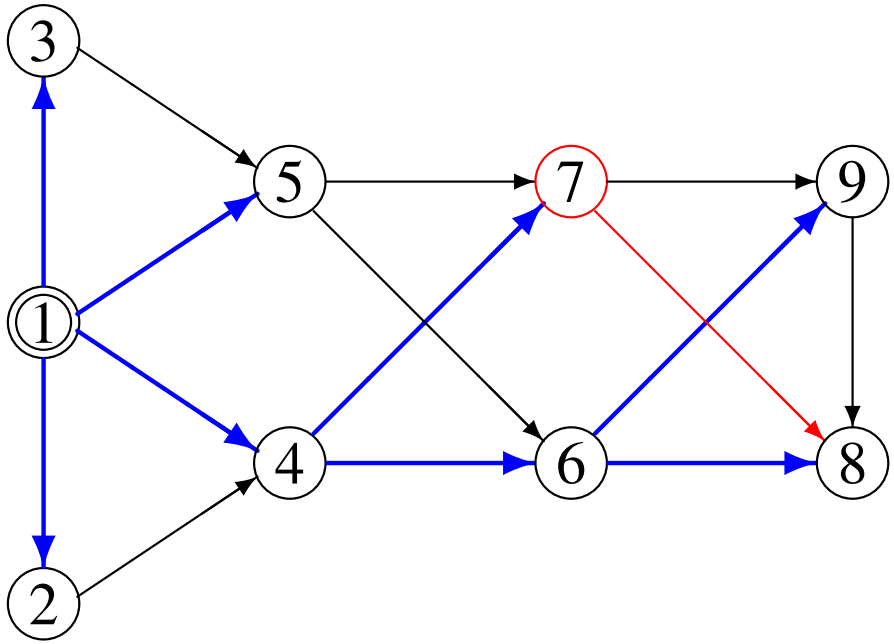
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
7	8,9,

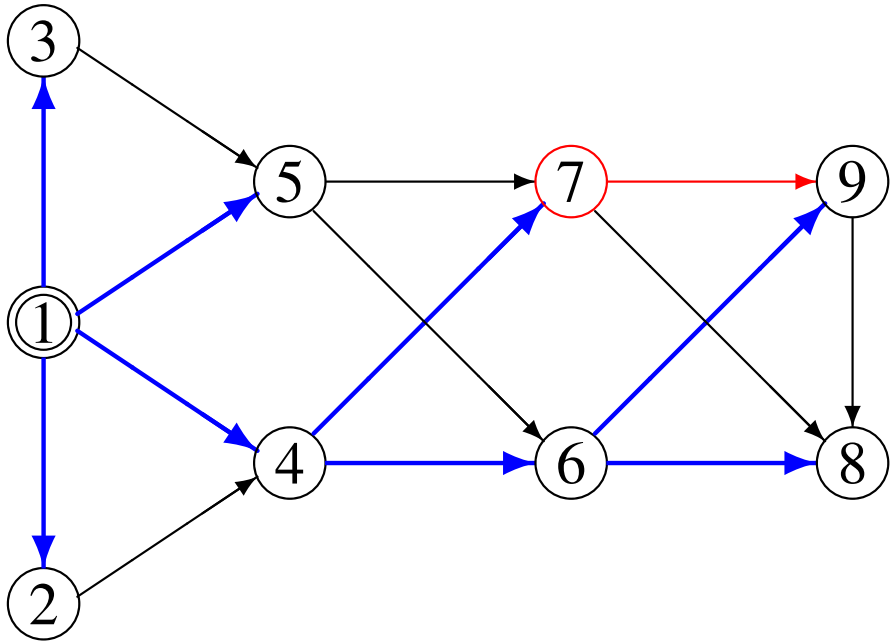
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
7	8,9,

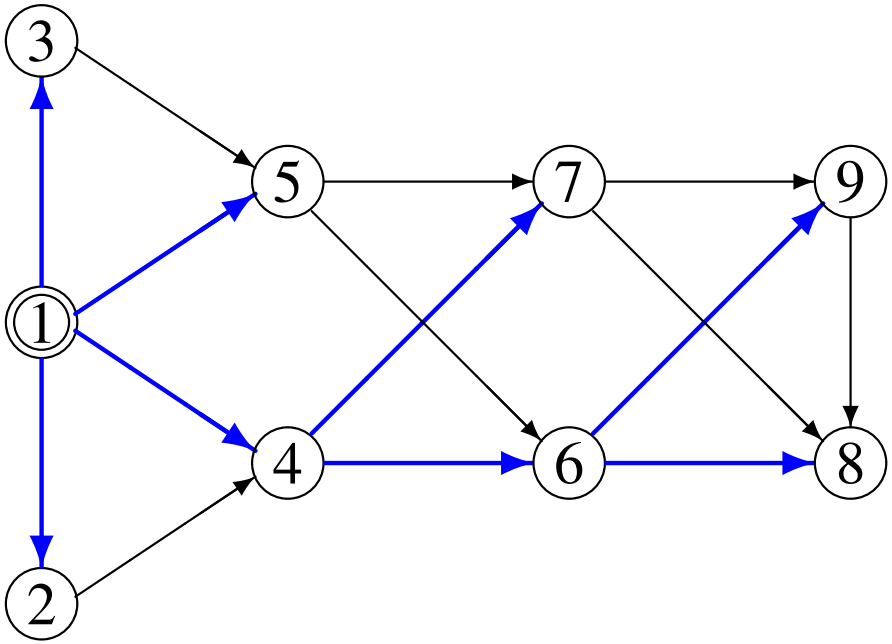
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
7	8,9,

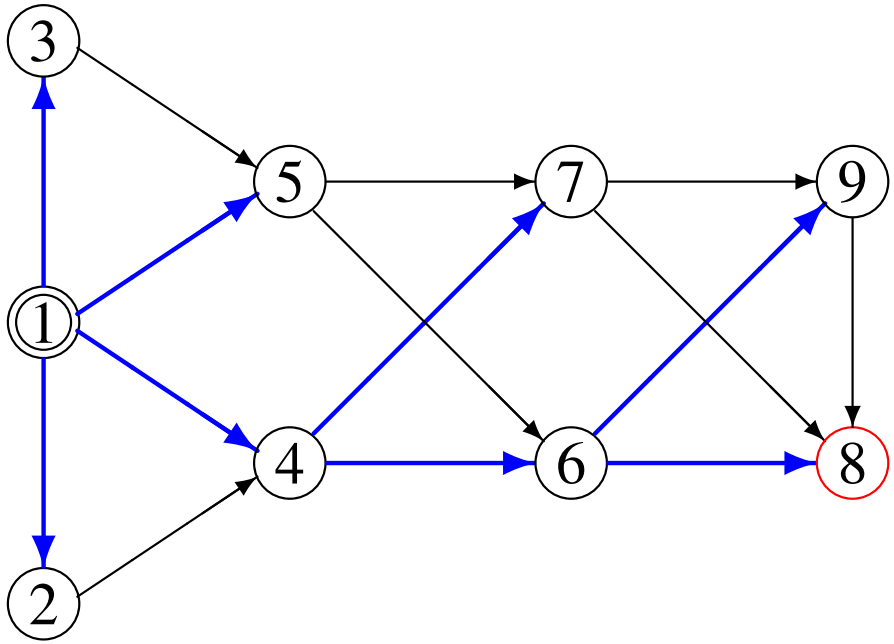
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
	8,9,

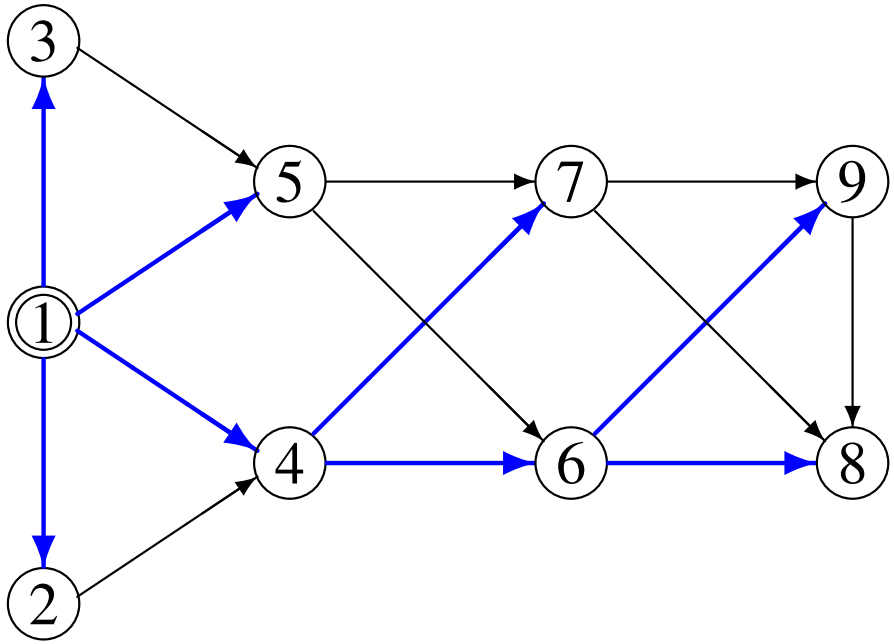
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
8	9,

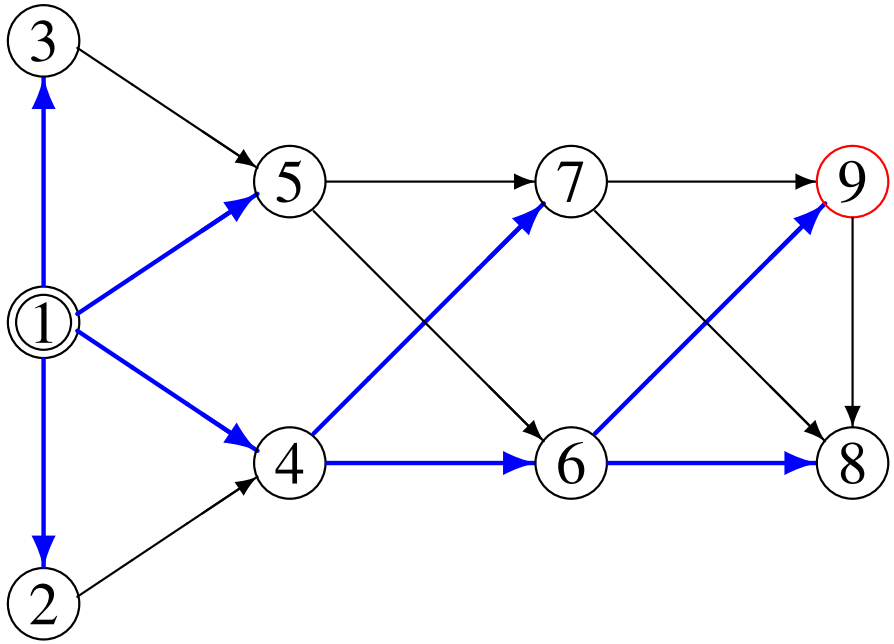
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
	9,

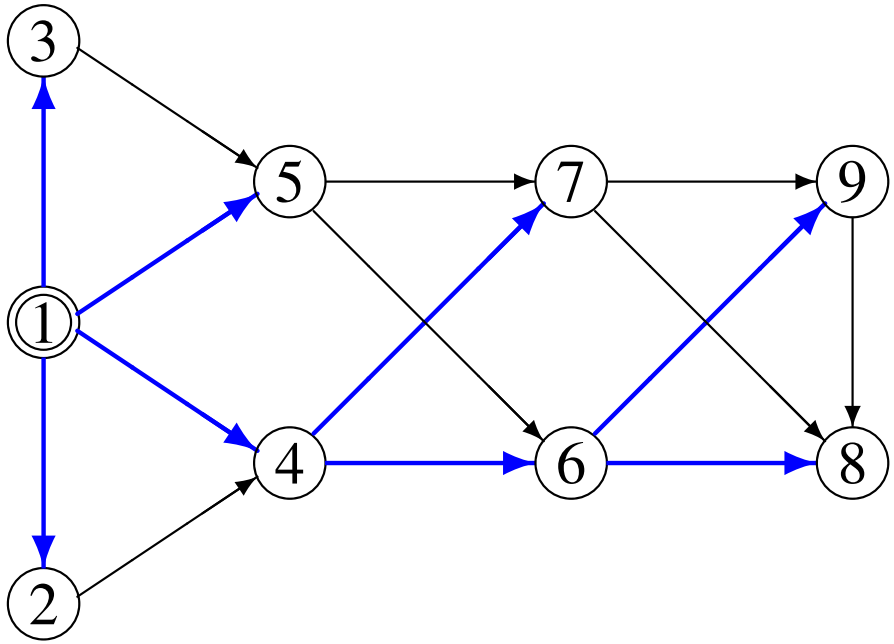
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q
9	

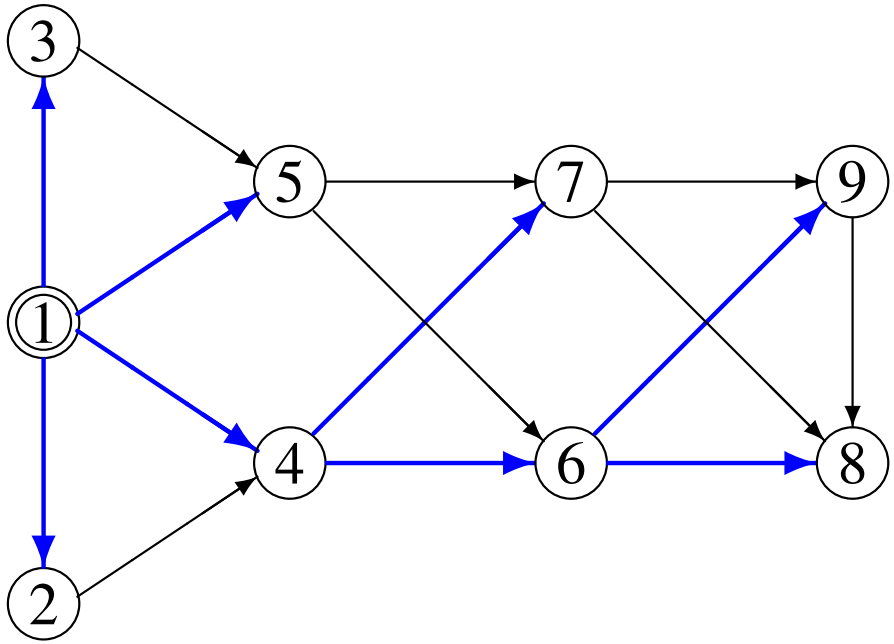
Пример поиска в ширину: $s = 1$



v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

v	Q

Пример поиска в ширину: $s = 1$



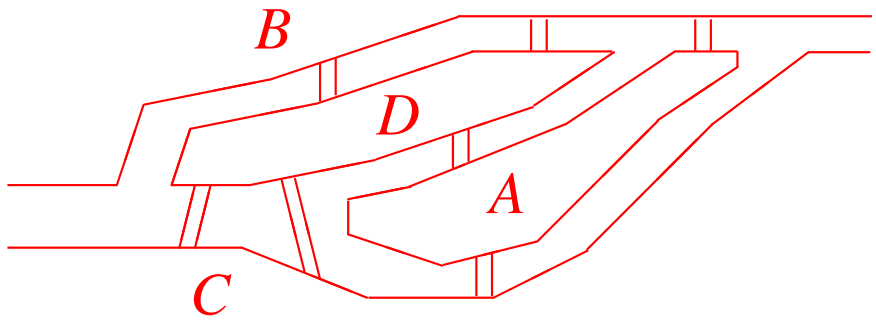
v	1	2	3	4	5	6	7	8	9
$parent(v)$	1	1	1	1	1	4	4	6	6

Очередь пуста $\Rightarrow$ указатели *parent* задают дерево поиска, дуги которого изображены синим цветом.

План лекции

Задача о кенигсбергских мостах

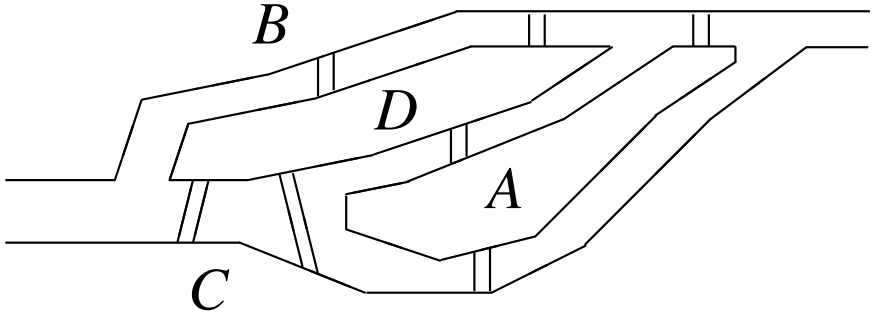
- На реке Преголь в Кенигсберге было два острова, которые соединялись между собой и с берегами реки семью мостами, как показано на рис. справа.



- Задача заключалась в том, чтобы, начав двигаться с одного из участков суши, помеченных на рисунке буквами A, B, C и D,
- пройти по каждому мосту ровно один раз и в результате вернуться в исходную точку.

Задача о кенигсбергских мостах

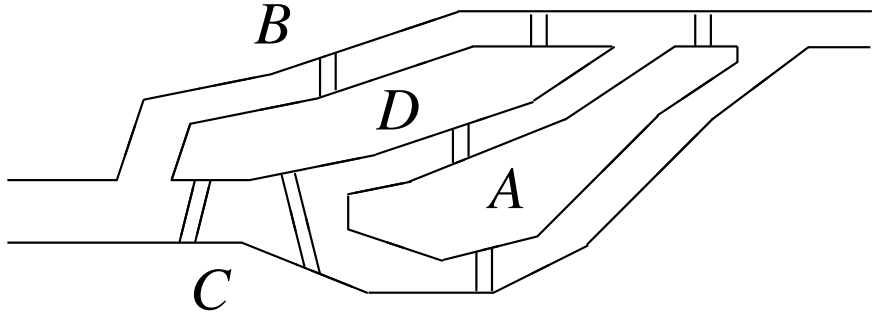
- На реке Преголь в Кенигсберге было два острова, которые соединялись между собой и с берегами реки семью мостами, как показано на рис. справа.



- Задача заключалась в том, чтобы, начав двигаться с одного из участков суши, помеченных на рисунке буквами A, B, C и D,
- пройти по каждому мосту ровно один раз и в результате вернуться в исходную точку.

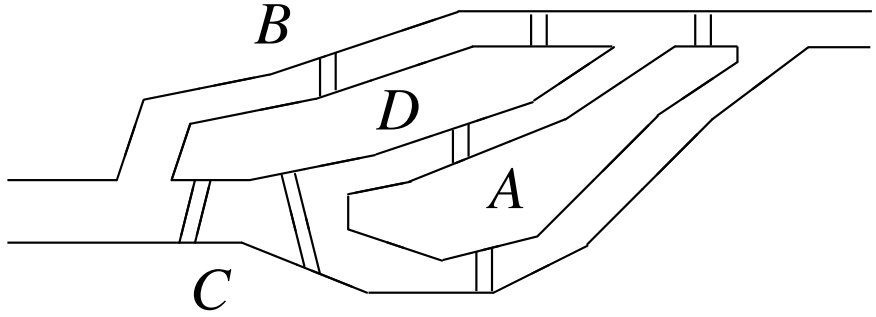
Задача о кенигсбергских мостах

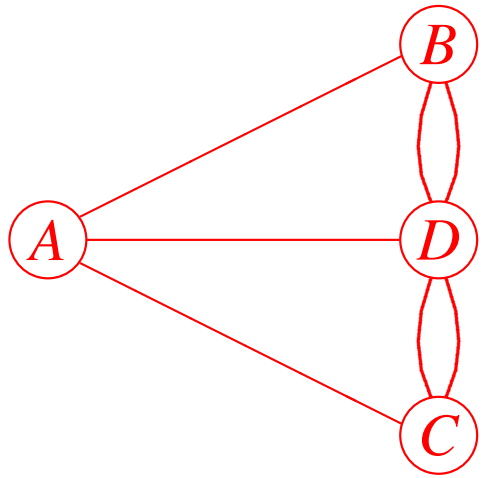
- На реке Преголь в Кенигсберге было два острова, которые соединялись между собой и с берегами реки семью мостами, как показано на рис. справа.
- Задача заключалась в том, чтобы, начав двигаться с одного из участков суши, помеченных на рисунке буквами A, B, C и D,
- пройти по каждому мосту ровно один раз и в результате вернуться в исходную точку.



Задача о кенигсбергских мостах

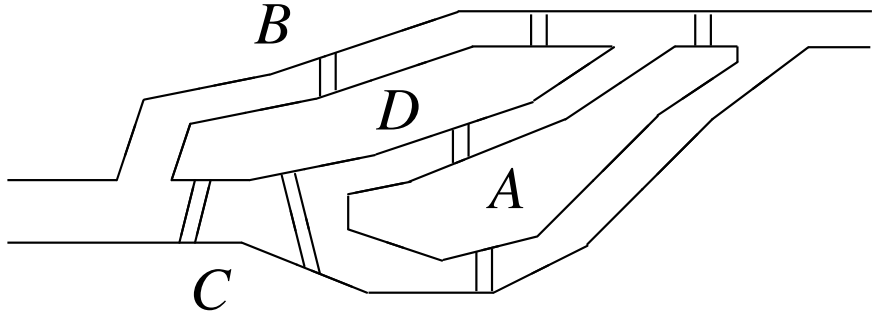
- В 1736 г. Эйлер доказал, что такого маршрута не существует, представив схему мостов мультиграфом.
- В этом мультиграфе нужно найти цикл (не обязательно простой), который проходит по каждому ребру ровно один раз.
- В последствии такие циклы стали называть *эйлеровыми*,
- а графы (мультиграфы), содержащие эйлеровы циклы, — *эйлеровыми графами* (*мультиграфами*).

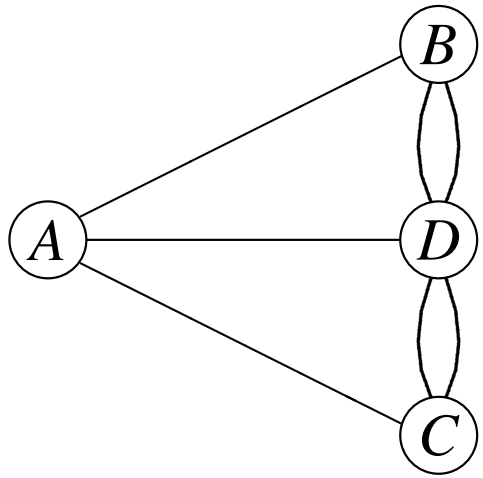




Задача о кенигсбергских мостах

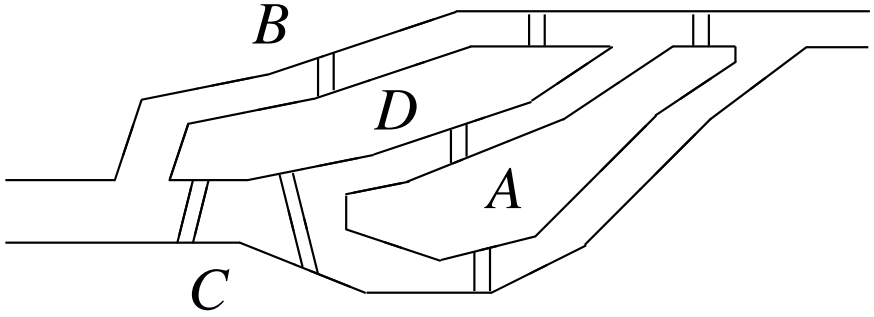
- В 1736 г. Эйлер доказал, что такого маршрута не существует, представив схему мостов мультиграфом.
- В этом мультиграфе нужно найти цикл (не обязательно простой), который проходит по каждому ребру ровно один раз.
- В последствии такие циклы стали называть *эйлеровыми*,
- а графы (мультиграфы), содержащие эйлеровы циклы, — *эйлеровыми графами* (*мультиграфами*).

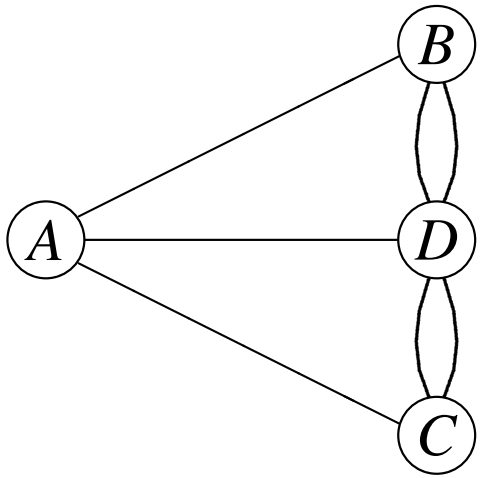




Задача о кенигсбергских мостах

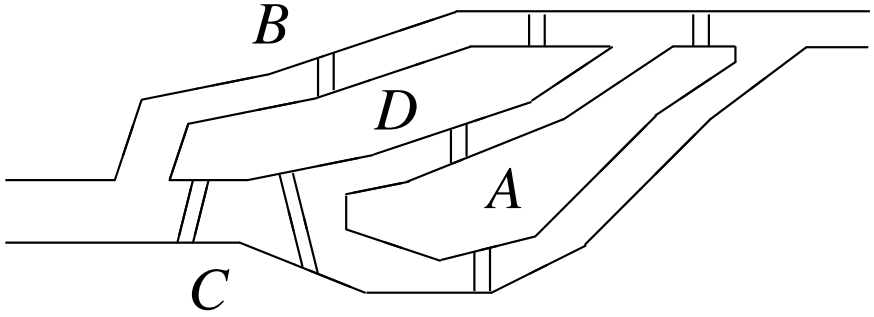
- В 1736 г. Эйлер доказал, что такого маршрута не существует, представив схему мостов мультиграфом.
- В этом мультиграфе нужно найти цикл (не обязательно простой), который проходит по каждому ребру ровно один раз.
- В последствии такие циклы стали называть *эйлеровыми*,
- а графы (мультиграфы), содержащие эйлеровы циклы, — *эйлеровыми графами* (*мультиграфами*).

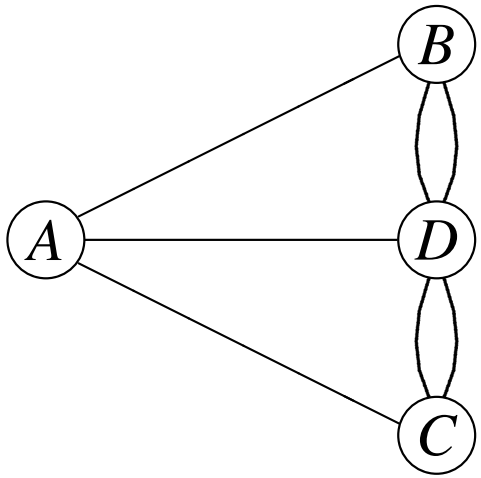




Задача о кенигсбергских мостах

- В 1736 г. Эйлер доказал, что такого маршрута не существует, представив схему мостов мультиграфом.
- В этом мультиграфе нужно найти цикл (не обязательно простой), который проходит по каждому ребру ровно один раз.
- В последствии такие циклы стали называть *эйлеровыми*,
- а графы (мультиграфы), содержащие эйлеровы циклы, — *эйлеровыми графами* (*мультиграфами*).





Характеризация эйлеровых графов

Теорема 2 (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен и степень каждой его вершины четная.

- Поскольку степень вершины A в мультиграфе Эйлера

- нечетна (равна трем),
- то этот мультиграф не имеет эйлерового цикла,
- и, следовательно, задача о кенигсбергских мостах также не имеет решения.

Характеризация эйлеровых графов

Теорема 2 (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен и степень каждой его вершины четная.

- Поскольку степень вершины A в мультиграфе Эйлера

- нечетна (равна трем),
- то этот мультиграф не имеет эйлерового цикла,
- и, следовательно, задача о кенигсбергских мостах также не имеет решения.

Характеризация эйлеровых графов

Теорема 2 (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен и степень каждой его вершины четная.

- Поскольку степень вершины A в мультиграфе Эйлера

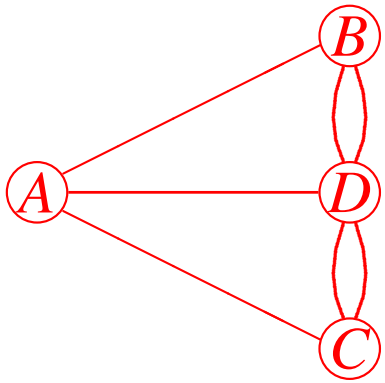
- нечетна (равна трем),
- то этот мультиграф не имеет эйлерового цикла,
- и, следовательно, задача о кенигсбергских мостах также не имеет решения.

Характеризация эйлеровых графов

Теорема 2 (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен и степень каждой его вершины четная.

- Поскольку степень вершины A в мультиграфе Эйлера



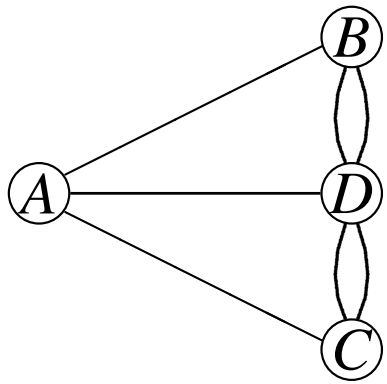
- нечетна (равна трем),
- то этот мультиграф не имеет эйлерового цикла,
- и, следовательно, задача о кенигсбергских мостах также не имеет решения.

Характеризация эйлеровых графов

Теорема 2 (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен и степень каждой его вершины четная.

- Поскольку степень вершины A в мультиграфе Эйлера



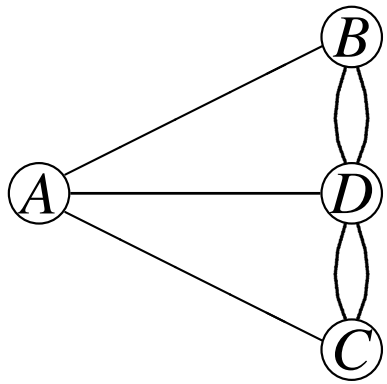
- нечетна (равна трем),
- то этот мультиграф не имеет эйлерового цикла,
- и, следовательно, задача о кенигсбергских мостах также не имеет решения.

Характеризация эйлеровых графов

Теорема 2 (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен и степень каждой его вершины четная.

- Поскольку степень вершины A в мультиграфе Эйлера



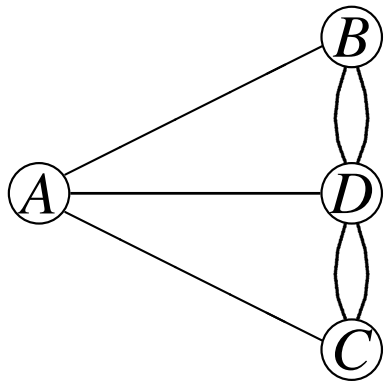
- нечетна (равна трем),
- то этот мультиграф не имеет эйлерового цикла,
- и, следовательно, задача о кенигсбергских мостах также не имеет решения.

Характеризация эйлеровых графов

Теорема 2 (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен и степень каждой его вершины четная.

- Поскольку степень вершины A в мультиграфе Эйлера



- нечетна (равна трем),
- то этот мультиграф не имеет эйлерового цикла,
- и, следовательно, задача о кенигсбергских мостах также не имеет решения.

Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.
- Простой цикл, содержащий все $n = |V|$ вершин графа, называется *гамильтоновым*.
- Граф G называется *гамильтоновым*, если он содержит гамильтонов цикл.
- Проверка того, что заданный граф является гамильтоновым, является одной из самых знаменитых задач теории графов.
- В 1859 г. известный математик У. Гамильтон головоломку, в которой требовалось найти обход всех вершин додекадрона, посещая каждую вершину не более одного раза.
- Эта задача эквивалентна задаче поиска гамильтонова цикла в следующем графе.

Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.
- Простой цикл, содержащий все $n = |V|$ вершин графа, называется *гамильтоновым*.
- Граф G называется *гамильтоновым*, если он содержит гамильтонов цикл.
- Проверка того, что заданный граф является гамильтоновым, является одной из самых знаменитых задач теории графов.
- В 1859 г. известный математик У. Гамильтон головоломку, в которой требовалось найти обход всех вершин додекадрона, посещая каждую вершину не более одного раза.
- Эта задача эквивалентна задаче поиска гамильтонова цикла в следующем графе.

Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.
- Простой цикл, содержащий все $n = |V|$ вершин графа, называется *гамильтоновым*.
- Граф G называется *гамильтоновым*, если он содержит гамильтонов цикл.
- Проверка того, что заданный граф является гамильтоновым, является одной из самых знаменитых задач теории графов.
- В 1859 г. известный математик У. Гамильтон головоломку, в которой требовалось найти обход всех вершин додекадрона, посещая каждую вершину не более одного раза.
- Эта задача эквивалентна задаче поиска гамильтонова цикла в следующем графе.

Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.
- Простой цикл, содержащий все $n = |V|$ вершин графа, называется *гамильтоновым*.
- Граф G называется *гамильтоновым*, если он содержит гамильтонов цикл.
- Проверка того, что заданный граф является гамильтоновым, является одной из самых знаменитых задач теории графов.
- В 1859 г. известный математик У. Гамильтон головоломку, в которой требовалось найти обход всех вершин додекадрона, посещая каждую вершину не более одного раза.
- Эта задача эквивалентна задаче поиска гамильтонова цикла в следующем графе.

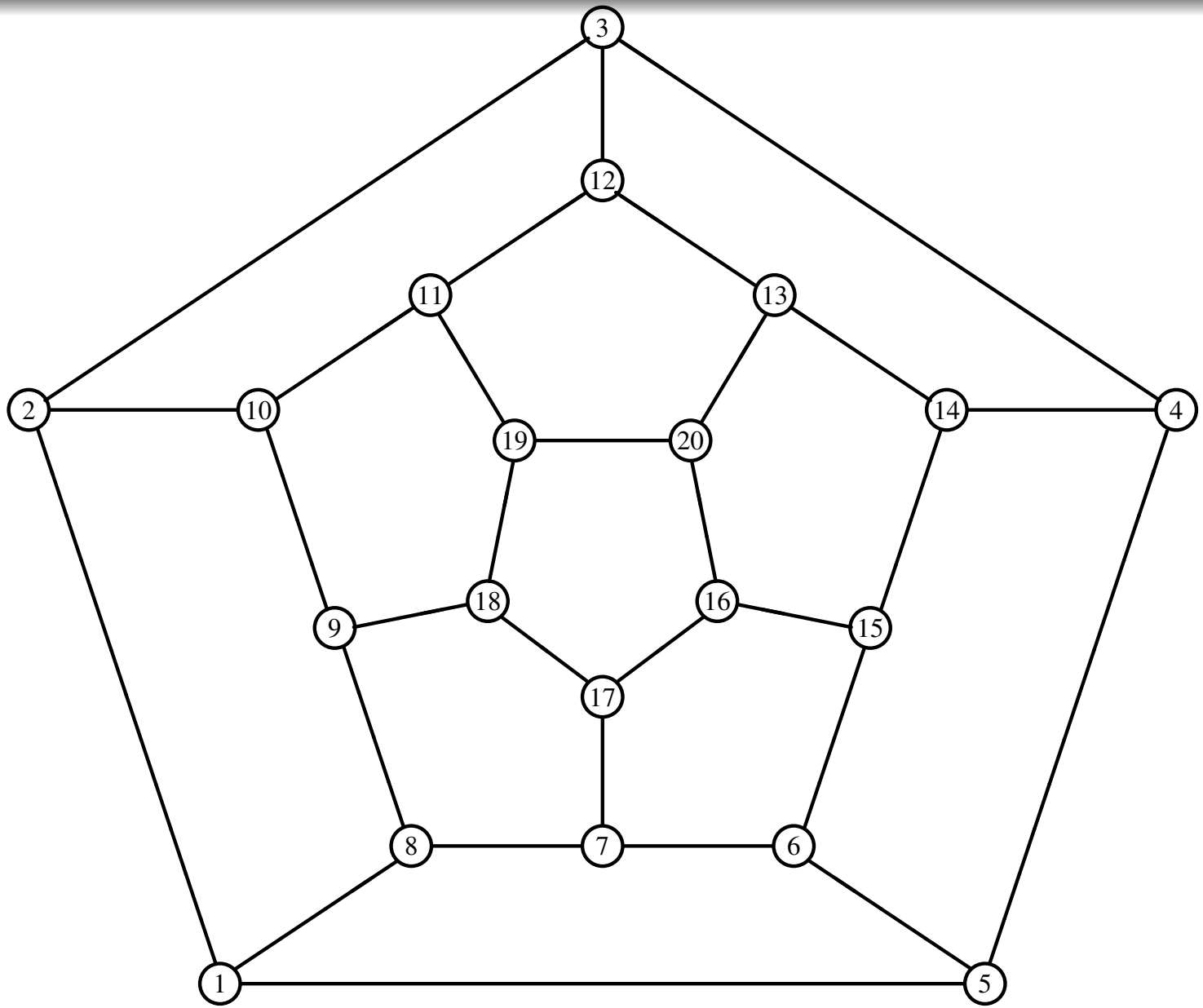
Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.
- Простой цикл, содержащий все $n = |V|$ вершин графа, называется *гамильтоновым*.
- Граф G называется *гамильтоновым*, если он содержит гамильтонов цикл.
- Проверка того, что заданный граф является гамильтоновым, является одной из самых знаменитых задач теории графов.
- В 1859 г. известный математик У. Гамильтон головоломку, в которой требовалось найти обход всех вершин додекадрона, посещая каждую вершину не более одного раза.
- Эта задача эквивалентна задаче поиска гамильтонова цикла в следующем графе.

Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.
- Простой цикл, содержащий все $n = |V|$ вершин графа, называется *гамильтоновым*.
- Граф G называется *гамильтоновым*, если он содержит гамильтонов цикл.
- Проверка того, что заданный граф является гамильтоновым, является одной из самых знаменитых задач теории графов.
- В 1859 г. известный математик У. Гамильтон головоломку, в которой требовалось найти обход всех вершин додекадрона, посещая каждую вершину не более одного раза.
- Эта задача эквивалентна задаче поиска гамильтонова цикла в следующем графе.

Граф головоломки Гамильтона



Задача Мерлина

- При дворе короля Артура проживали 500 рыцарей, не все из которых ладили между собой.
- Из-за этого во время обеда, где все рыцари сидели за круглым столом, часто возникали драки между сидящими рядом рыцарями.
- Королю Артуру это не нравилось, и он приказал своему магу Мерлину рассадить рыцарей таким образом, чтобы рядом не сидели враждующие рыцари.
- Задача Мерлина формулируется как задача поиска гамильтонова цикла в графе, в котором
- вершины представляют рыцарей,
- и две вершины соединены ребром, если соответствующие им рыцари не враждуют между собой.

Задача Мерлина

- При дворе короля Артура проживали 500 рыцарей, не все из которых ладили между собой.
- Из-за этого во время обеда, где все рыцари сидели за круглым столом, часто возникали драки между сидящими рядом рыцарями.
- Королю Артуру это не нравилось, и он приказал своему магу Мерлину рассадить рыцарей таким образом, чтобы рядом не сидели враждующие рыцари.
- Задача Мерлина формулируется как задача поиска гамильтонова цикла в графе, в котором
- вершины представляют рыцарей,
- и две вершины соединены ребром, если соответствующие им рыцари не враждуют между собой.

Задача Мерлина

- При дворе короля Артура проживали 500 рыцарей, не все из которых ладили между собой.
- Из-за этого во время обеда, где все рыцари сидели за круглым столом, часто возникали драки между сидящими рядом рыцарями.
- Королю Артуру это не нравилось, и он приказал своему магу Мерлину рассадить рыцарей таким образом, чтобы рядом не сидели враждующие рыцари.
- Задача Мерлина формулируется как задача поиска гамильтонова цикла в графе, в котором
- вершины представляют рыцарей,
- и две вершины соединены ребром, если соответствующие им рыцари не враждуют между собой.

Задача Мерлина

- При дворе короля Артура проживали 500 рыцарей, не все из которых ладили между собой.
- Из-за этого во время обеда, где все рыцари сидели за круглым столом, часто возникали драки между сидящими рядом рыцарями.
- Королю Артуру это не нравилось, и он приказал своему магу Мерлину рассадить рыцарей таким образом, чтобы рядом не сидели враждующие рыцари.
- Задача Мерлина формулируется как задача поиска гамильтонова цикла в графе, в котором
- вершины представляют рыцарей,
- и две вершины соединены ребром, если соответствующие им рыцари не враждуют между собой.

Задача Мерлина

- При дворе короля Артура проживали 500 рыцарей, не все из которых ладили между собой.
- Из-за этого во время обеда, где все рыцари сидели за круглым столом, часто возникали драки между сидящими рядом рыцарями.
- Королю Артуру это не нравилось, и он приказал своему магу Мерлину рассадить рыцарей таким образом, чтобы рядом не сидели враждующие рыцари.
- Задача Мерлина формулируется как задача поиска гамильтонова цикла в графе, в котором
- вершины представляют рыцарей,
- и две вершины соединены ребром, если соответствующие им рыцари не враждуют между собой.

Задача Мерлина

- При дворе короля Артура проживали 500 рыцарей, не все из которых ладили между собой.
- Из-за этого во время обеда, где все рыцари сидели за круглым столом, часто возникали драки между сидящими рядом рыцарями.
- Королю Артуру это не нравилось, и он приказал своему магу Мерлину рассадить рыцарей таким образом, чтобы рядом не сидели враждующие рыцари.
- Задача Мерлина формулируется как задача поиска гамильтонова цикла в графе, в котором
- вершины представляют рыцарей,
- и две вершины соединены ребром, если соответствующие им рыцари не враждуют между собой.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

Задача о минимальном гамильтоновом цикле

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписана стоимость $c(v, w)$,
- и нам нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- минимальной стоимости $c(\Gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- *Задача коммивояжера* — это задача о минимальном гамильтоновом цикле в полном графе.
- Вершины графа представляют некоторые города, а стоимости $c(v, w)$ — это расстояния между городами.
- Коммивояжер, начиная из города, в котором он проживает,
- хочет посетить каждый из остальных $n - 1$ городов ровно один раз и вернуться обратно в родной город,
- при этом длина его маршрута должна быть минимальной.

План лекции

Задача о максимальной клике

- Граф $H = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$,
- если $\bar{V} \subseteq V$ и $\bar{E} \subseteq E$.
- Максимальный (по включению) полный подграф графа G называется *кликой*.
- На практике часто встречается *задача о максимальной клике*, целью в которой является поиск клики с максимальным количеством вершин.
- Для примера, пусть вершины графа представляют некоторую группу людей, и две вершины соединены ребром, если соответствующие им люди знакомы друг с другом.
- Мы решаем задачу о максимальной клике, когда ходим найти наибольшую подгруппу людей попарно знакомых друг с другом

Задача о максимальной клике

- Граф $H = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$,
- если $\bar{V} \subseteq V$ и $\bar{E} \subseteq E$.
- Максимальный (по включению) полный подграф графа G называется *кликой*.
- На практике часто встречается *задача о максимальной клике*, целью в которой является поиск клики с максимальным количеством вершин.
- Для примера, пусть вершины графа представляют некоторую группу людей, и две вершины соединены ребром, если соответствующие им люди знакомы друг с другом.
- Мы решаем задачу о максимальной клике, когда ходим найти наибольшую подгруппу людей попарно знакомых друг с другом

Задача о максимальной клике

- Граф $H = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$,
- если $\bar{V} \subseteq V$ и $\bar{E} \subseteq E$.
- Максимальный (по включению) полный подграф графа G называется *кликой*.
- На практике часто встречается *задача о максимальной клике*, целью в которой является поиск клики с максимальным количеством вершин.
- Для примера, пусть вершины графа представляют некоторую группу людей, и две вершины соединены ребром, если соответствующие им люди знакомы друг с другом.
- Мы решаем задачу о максимальной клике, когда ходим найти наибольшую подгруппу людей попарно знакомых друг с другом

Задача о максимальной клике

- Граф $H = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$,
- если $\bar{V} \subseteq V$ и $\bar{E} \subseteq E$.
- Максимальный (по включению) полный подграф графа G называется *кликой*.
- На практике часто встречается *задача о максимальной клике*, целью в которой является поиск клики с максимальным количеством вершин.
- Для примера, пусть вершины графа представляют некоторую группу людей, и две вершины соединены ребром, если соответствующие им люди знакомы друг с другом.
- Мы решаем задачу о максимальной клике, когда ходим найти наибольшую подгруппу людей попарно знакомых друг с другом

Задача о максимальной клике

- Граф $H = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$,
- если $\bar{V} \subseteq V$ и $\bar{E} \subseteq E$.
- Максимальный (по включению) полный подграф графа G называется *кликой*.
- На практике часто встречается *задача о максимальной клике*, целью в которой является поиск клики с максимальным количеством вершин.
- Для примера, пусть вершины графа представляют некоторую группу людей, и две вершины соединены ребром, если соответствующие им люди знакомы друг с другом.
- Мы решаем задачу о максимальной клике, когда ходим найти наибольшую подгруппу людей попарно знакомых друг с другом

Задача о максимальной клике

- Граф $H = (\bar{V}, \bar{E})$ называется *подграфом* графа $G = (V, E)$,
- если $\bar{V} \subseteq V$ и $\bar{E} \subseteq E$.
- Максимальный (по включению) полный подграф графа G называется *кликой*.
- На практике часто встречается *задача о максимальной клике*, целью в которой является поиск клики с максимальным количеством вершин.
- Для примера, пусть вершины графа представляют некоторую группу людей, и две вершины соединены ребром, если соответствующие им люди знакомы друг с другом.
- Мы решаем задачу о максимальной клике, когда ходим найти наибольшую подгруппу людей попарно знакомых друг с другом

Раскраска графа и карт

- В какое минимальное число цветов можно раскрасить вершины заданного графа, чтобы никакие две смежные вершины не были окрашены в один цвет.
- Так формулируется *задача о раскраске графа*.
- Самый знаменитый частный случай данной задачи, известный как *проблема четырех красок*, состоит в том,
- чтобы определить минимальное число цветов, необходимых для раскраски политической карты так,
- чтобы никакие две страны, имеющие общую границу, не были раскрашены в один цвет.
- Если представить каждую страну отдельной вершиной графа и соединить две вершины ребром, если соответствующие им страны имеют общую границу,
- то задача о раскраске карты представляется как задача о раскраске полученного графа.

Раскраска графа и карт

- В какое минимальное число цветов можно раскрасить вершины заданного графа, чтобы никакие две смежные вершины не были окрашены в один цвет.
- Так формулируется *задача о раскраске графа.*
- Самый знаменитый частный случай данной задачи, известный как *проблема четырех красок*, состоит в том,
- чтобы определить минимальное число цветов, необходимых для раскраски политической карты так,
- чтобы никакие две страны, имеющие общую границу, не были раскрашены в один цвет.
- Если представить каждую страну отдельной вершиной графа и соединить две вершины ребром, если соответствующие им страны имеют общую границу,
- то задача о раскраске карты представляется как задача о раскраске полученного графа.

Раскраска графа и карт

- В какое минимальное число цветов можно раскрасить вершины заданного графа, чтобы никакие две смежные вершины не были окрашены в один цвет.
- Так формулируется *задача о раскраске графа*.
- Самый знаменитый частный случай данной задачи, известный как *проблема четырех красок*, состоит в том,
- чтобы определить минимальное число цветов, необходимых для раскраски политической карты так,
- чтобы никакие две страны, имеющие общую границу, не были раскрашены в один цвет.
- Если представить каждую страну отдельной вершиной графа и соединить две вершины ребром, если соответствующие им страны имеют общую границу,
- то задача о раскраске карты представляется как задача о раскраске полученного графа.

Раскраска графа и карт

- В какое минимальное число цветов можно раскрасить вершины заданного графа, чтобы никакие две смежные вершины не были окрашены в один цвет.
- Так формулируется *задача о раскраске графа*.
- Самый знаменитый частный случай данной задачи, известный как *проблема четырех красок*, состоит в том,
- чтобы определить минимальное число цветов, необходимых для раскраски политической карты так,
- чтобы никакие две страны, имеющие общую границу, не были раскрашены в один цвет.
- Если представить каждую страну отдельной вершиной графа и соединить две вершины ребром, если соответствующие им страны имеют общую границу,
- то задача о раскраске карты представляется как задача о раскраске полученного графа.

Раскраска графа и карт

- В какое минимальное число цветов можно раскрасить вершины заданного графа, чтобы никакие две смежные вершины не были окрашены в один цвет.
- Так формулируется *задача о раскраске графа*.
- Самый знаменитый частный случай данной задачи, известный как *проблема четырех красок*, состоит в том,
- чтобы определить минимальное число цветов, необходимых для раскраски политической карты так,
- чтобы никакие две страны, имеющие общую границу, не были раскрашены в один цвет.
- Если представить каждую страну отдельной вершиной графа и соединить две вершины ребром, если соответствующие им страны имеют общую границу,
- то задача о раскраске карты представляется как задача о раскраске полученного графа.

Раскраска графа и карт

- В какое минимальное число цветов можно раскрасить вершины заданного графа, чтобы никакие две смежные вершины не были окрашены в один цвет.
- Так формулируется *задача о раскраске графа*.
- Самый знаменитый частный случай данной задачи, известный как *проблема четырех красок*, состоит в том,
- чтобы определить минимальное число цветов, необходимых для раскраски политической карты так,
- чтобы никакие две страны, имеющие общую границу, не были раскрашены в один цвет.
- Если представить каждую страну отдельной вершиной графа и соединить две вершины ребром, если соответствующие им страны имеют общую границу,
- то задача о раскраске карты представляется как задача о раскраске полученного графа.

Раскраска графа и карт

- В какое минимальное число цветов можно раскрасить вершины заданного графа, чтобы никакие две смежные вершины не были окрашены в один цвет.
- Так формулируется *задача о раскраске графа*.
- Самый знаменитый частный случай данной задачи, известный как *проблема четырех красок*, состоит в том,
- чтобы определить минимальное число цветов, необходимых для раскраски политической карты так,
- чтобы никакие две страны, имеющие общую границу, не были раскрашены в один цвет.
- Если представить каждую страну отдельной вершиной графа и соединить две вершины ребром, если соответствующие им страны имеют общую границу,
- то задача о раскраске карты представляется как задача о раскраске полученного графа.

Проблема четырех красок

- Нетрудно привести пример карты, для раскраски которой требуется четыре цвета.
- Долгое время гипотеза о том, что четырех цветов достаточно для раскраски любой карты оставалась недоказанной. Это было сделано Аппелем и Хакеном в 1976 г. (K.I. Appel, W. Haken. Every planar map is four-colorable. *Bull. Am. Math. Soc.* **82** (1976) 711–712.) оригинальным способом:
- сначала доказательство гипотезы было сведено к рассмотрению достаточно большого числа частных случаев задачи,
- а затем была написана компьютерная программа, которая выполнила «раскраску» карт для каждого из выделенных случаев.

Проблема четырех красок

- Нетрудно привести пример карты, для раскраски которой требуется четыре цвета.
- Долгое время гипотеза о том, что четырех цветов достаточно для раскраски любой карты оставалась недоказанной. Это было сделано Аппелем и Хакеном в 1976 г. (K.I. Appel, W. Haken. Every planar map is four-colorable. *Bull. Am. Math. Soc.* **82** (1976) 711–712.) оригинальным способом:
- сначала доказательство гипотезы было сведено к рассмотрению достаточно большого числа частных случаев задачи,
- а затем была написана компьютерная программа, которая выполнила «раскраску» карт для каждого из выделенных случаев.

Проблема четырех красок

- Нетрудно привести пример карты, для раскраски которой требуется четыре цвета.
- Долгое время гипотеза о том, что четырех цветов достаточно для раскраски любой карты оставалась недоказанной. Это было сделано Аппелем и Хакеном в 1976 г. (K.I. Appel, W. Haken. Every planar map is four-colorable. *Bull. Am. Math. Soc.* **82** (1976) 711–712.) оригинальным способом:
- сначала доказательство гипотезы было сведено к рассмотрению достаточно большого числа частных случаев задачи,
- а затем была написана компьютерная программа, которая выполнила «раскраску» карт для каждого из выделенных случаев.

Проблема четырех красок

- Нетрудно привести пример карты, для раскраски которой требуется четыре цвета.
- Долгое время гипотеза о том, что четырех цветов достаточно для раскраски любой карты оставалась недоказанной. Это было сделано Аппелем и Хакеном в 1976 г. (K.I. Appel, W. Haken. Every planar map is four-colorable. *Bull. Am. Math. Soc.* **82** (1976) 711–712.) оригинальным способом:
- сначала доказательство гипотезы было сведено к рассмотрению достаточно большого числа частных случаев задачи,
- а затем была написана компьютерная программа, которая выполнила «раскраску» карт для каждого из выделенных случаев.

Укладка графа на плоскости

- Как мы уже видели, графы можно рисовать на плоскости, причем, это можно сделать разными способами.
- Считается, что рисунок графа более привлекателен, если на нем количество пересечений ребер минимально.
- В идеале, хотелось бы полностью избежать пересечений ребер, но это не всегда возможно.
- Два самых «маленьких» графа, которые нельзя нарисовать на плоскости без пересечений ребер, — это графы K_5 и $K_{3,3}$.

Укладка графа на плоскости

- Как мы уже видели, графы можно рисовать на плоскости, причем, это можно сделать разными способами.
- Считается, что рисунок графа более привлекателен, если на нем количество пересечений ребер минимально.
- В идеале, хотелось бы полностью избежать пересечений ребер, но это не всегда возможно.
- Два самых «маленьких» графа, которые нельзя нарисовать на плоскости без пересечений ребер, — это графы K_5 и $K_{3,3}$.

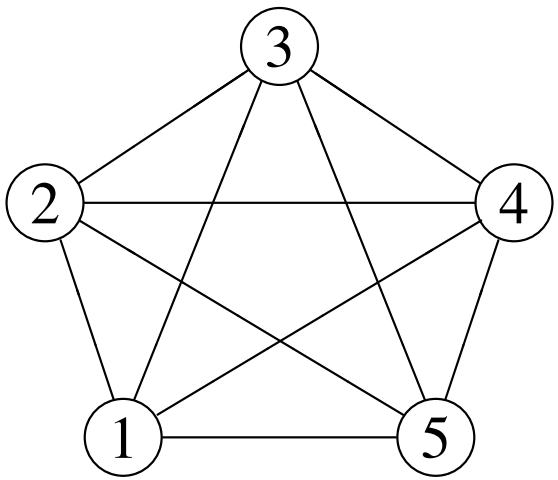
Укладка графа на плоскости

- Как мы уже видели, графы можно рисовать на плоскости, причем, это можно сделать разными способами.
- Считается, что рисунок графа более привлекателен, если на нем количество пересечений ребер минимально.
- В идеале, хотелось бы полностью избежать пересечений ребер, но это не всегда возможно.
- Два самых «маленьких» графа, которые нельзя нарисовать на плоскости без пересечений ребер, — это графы K_5 и $K_{3,3}$.

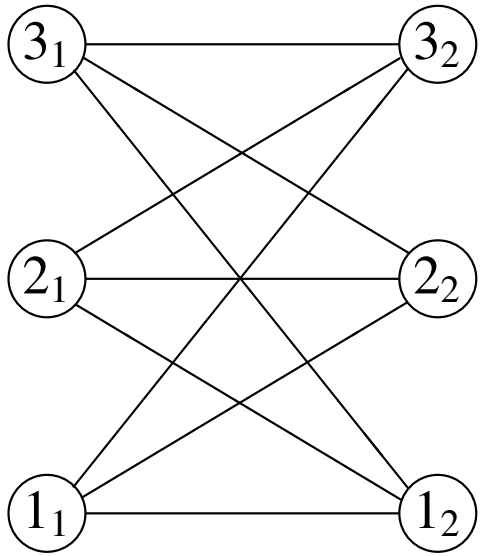
Укладка графа на плоскости

- Как мы уже видели, графы можно рисовать на плоскости, причем, это можно сделать разными способами.
- Считается, что рисунок графа более привлекателен, если на нем количество пересечений ребер минимально.
- В идеале, хотелось бы полностью избежать пересечений ребер, но это не всегда возможно.
- Два самых «маленьких» графа, которые нельзя нарисовать на плоскости без пересечений ребер, — это графы K_5 и $K_{3,3}$.

Примеры непланарных графов



граф K_5



граф $K_{3,3}$

Укладка графа на плоскости

- Граф, который можно нарисовать на плоскости без пересечения ребер, называется *планарным*.
- Если граф G непланарен, то также непланарен и граф G', который получается из исходного переименованием вершин и заменой нескольких его ребер простыми путями.
- Графы G и G' называются *гомеоморфными*.

Теорема 3 (Куратовского)

Граф G планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 и $K_{3,3}$.

Укладка графа на плоскости

- Граф, который можно нарисовать на плоскости без пересечения ребер, называется *планарным*.
- Если граф G непланарен, то также непланарен и граф G', который получается из исходного переименованием вершин и заменой нескольких его ребер простыми путями.
- Графы G и G' называются *гомеоморфными*.

Теорема 3 (Куратовского)

Граф G планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 и $K_{3,3}$.

Укладка графа на плоскости

- Граф, который можно нарисовать на плоскости без пересечения ребер, называется *планарным*.
- Если граф G непланарен, то также непланарен и граф G', который получается из исходного переименованием вершин и заменой нескольких его ребер простыми путями.
- Графы G и G' называются *гомеоморфными*.

Теорема 3 (Куратовского)

Граф G планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 и $K_{3,3}$.

Укладка графа на плоскости

- Граф, который можно нарисовать на плоскости без пересечения ребер, называется *планарным*.
- Если граф G непланарен, то также непланарен и граф G', который получается из исходного переименованием вершин и заменой нескольких его ребер простыми путями.
- Графы G и G' называются *гомеоморфными*.

Теорема 3 (Куратовского)

Граф G планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 и $K_{3,3}$.

Укладка графа на плоскости

- Граф, который можно нарисовать на плоскости без пересечения ребер, называется *планарным*.
- Если граф G непланарен, то также непланарен и граф G', который получается из исходного переименованием вершин и заменой нескольких его ребер простыми путями.
- Графы G и G' называются *гомеоморфными*.

Теорема 3 (Куратовского)

Граф G планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 и $K_{3,3}$.

Укладка графа на плоскости

- Граф, который можно нарисовать на плоскости без пересечения ребер, называется *планарным*.
- Если граф G непланарен, то также непланарен и граф G', который получается из исходного переименованием вершин и заменой нескольких его ребер простыми путями.
- Графы G и G' называются *гомеоморфными*.

Теорема 3 (Куратовского)

Граф G планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 и $K_{3,3}$.