Графы

Виктор Васильевич Лепин

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 $G = (V, E)$,

Неориентированные графы

- Графом называется пара $G = (V, E)$,
- где V – конечное множество, элементы которого называются вершинами,

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.

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.

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.

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.

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: $\sum_{v \in V} deg(v) = 2|E|$.

Ориентированные графы

- Ориентированным графом (орграфом) называется пара $G = (V, E)$,

Ориентированные графы

- Ориентированным графом (орграфом) называется пара $G = (V, E)$,
- где V – конечное множество вершин,

Ориентированные графы

- Ориентированным графом (орграфом) называется пара $G = (V, E)$,
- где V – конечное множество вершин,
- а E – это множество упорядоченных пар вершин.

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.

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.

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 $outdeg(v)$, называется количество дуг, выходящих из 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 $outdeg(v)$, называется количество дуг, выходящих из v.
- Степенью захода вершины v, обозначается $indeg(v)$, называется количество дуг, входящих в v.

Мультиграфы

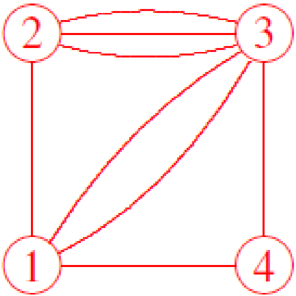
Иногда полезно рассматривать мультиграфы,

Мультиграфы

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

Мультиграфы

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



Подграфы

- Граф $G' = (V', E')$ называется подграфом графа $G = (V, E)$, если
  - $V' \subseteq V$,
  - $E' \subseteq E$,
- Подграфом графа $G = (V, E)$, порожденным множеством вершин $U \subseteq V$, называется подграф $G(U) = (U, E(U))$,

Подграфы

- Граф $G' = (V', E')$ называется подграфом графа $G = (V, E)$, если
  - $V' \subseteq V$,
  - $E' \subseteq E$,
- Подграфом графа $G = (V, E)$, порожденным множеством вершин $U \subseteq V$, называется подграф $G(U) = (U, E(U))$,
- где $E(U) = \{(v, w) \in E : v \in U, w \in U\}$.

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) $G = (V, E)$,

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 $i = 1, \ldots, k$.

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.

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.

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.

Основные определения

Вершина (vertex, node)

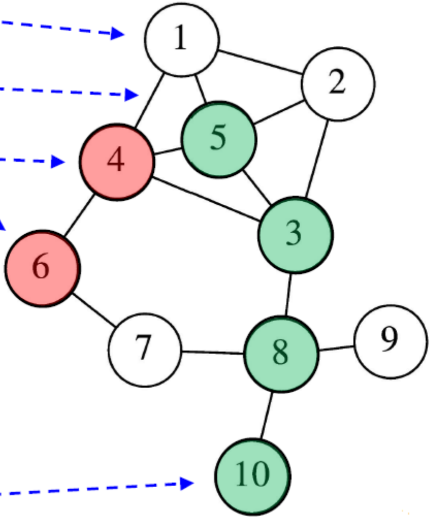
Ребро (edge, link)

Смежные вершины
(adjacent vertices)

Ребро (4, 6) *инцидентно*
(incident) вершинам 4 и 6

Путь (path) – последовательность вершин, в которой следующая вершина (после первой) является смежной с предыдущей
(все вершины и ребра в пути различны)

Путь: (10, 8, 3, 5)



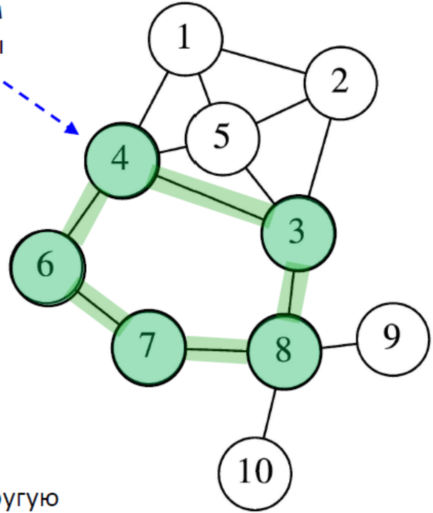
Основные определения

Цикл (cycle) – путь, в котором первая и последняя вершины совпадают: (4, 6, 7, 8, 3, 4)

Степень вершины (vertex degree) – количество ребер, инцидентных вершине

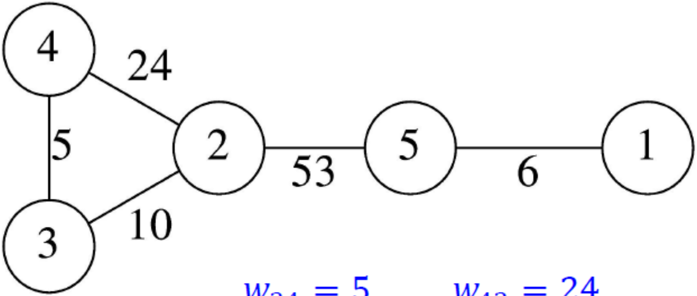
deg(7) = 2,   deg(1) = 3

Связный граф (connected graph) – граф, в котором существует путь из каждой вершины в любую другую



Основные определения

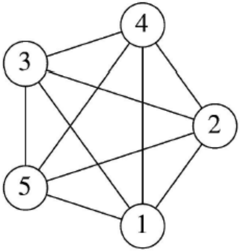
- **Взвешенный граф** (weighted graph) – это граф, ребрами (дугам) которого назначены веса
- Вес ребра (i, j) обозначим как w_{ij}



$w_{34} = 5, \quad w_{42} = 24, \quad \ldots$

Основные определения

Полный граф (complete graph) – это граф, в котором каждая пара различных вершин смежна (каждая вершина соединена со всеми)

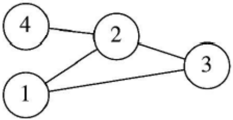


Количество ребер в полном неориентированном графе:

$$m = \frac{n(n-1)}{2}$$

Насыщенность *D* графа (density):

$$D = \frac{2m}{n(n-1)}$$



$$D = \frac{4}{\frac{4 \cdot 3}{2}} = \frac{4}{6} = 0.66$$

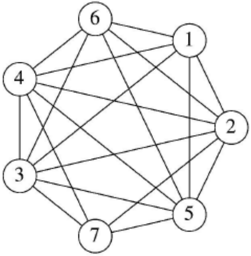
У полного графа насыщенность *D* = 1

Основные определения

Насыщенный граф (dense graph) – это граф, в котором количество ребер близко к максимально возможному

$$|E| = O(|V|^2)$$

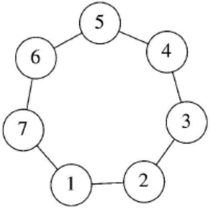
$$D = \frac{2 \cdot 19}{7 \cdot 6} = 0.9, \quad D > 0.5$$



Разреженный граф (sparse graph) – граф, в котором количество ребер близко к количеству вершин в графе

$$|E| = O(|V|)$$

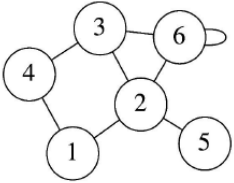
$$D = \frac{2 \cdot 7}{7 \cdot 6} = 0.33, \quad D < 0.5$$



Матрица смежности

- **Матрица *A* смежности** (adjacency matrix) – это матрица $n \times n$ элементов, в которой

$$a_{ij} = \begin{cases} 1, & \text{если } (i,j) \in E, \\ 0, & \text{иначе.} \end{cases}$$



- Объем требуемой памяти $O(|V|^2)$
- Быстрое определение присутствия ребра (i,j) в графе
- За время $O(1)$ получаем доступ к элементу a_{ij} матрицы

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Эффективна для насыщенных графов ($|E| \approx |V|^2$)

Матрица смежности

- **Какого размера граф можно разместить в оперативной памяти объемом 8 Гб использую матрицу смежности?**

```
int a[n][n];
```

Матрица смежности

- **Какого размера граф можно разместить в оперативной памяти объемом 8 Гб использую матрицу смежности?**

```
int a[n][n];
```

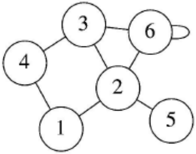
- sizeof(int) = 4 байта
- 8 Гб = 8 · 2^{30} байт
- 8 · 2^{30} / 4 = 2 · 2^{30} – можно разместить 2^{31} = 2 147 483 648 элементов типа int
- $n = \lfloor\sqrt{2^{31}}\rfloor = 46340$ – количество строки и столбцов

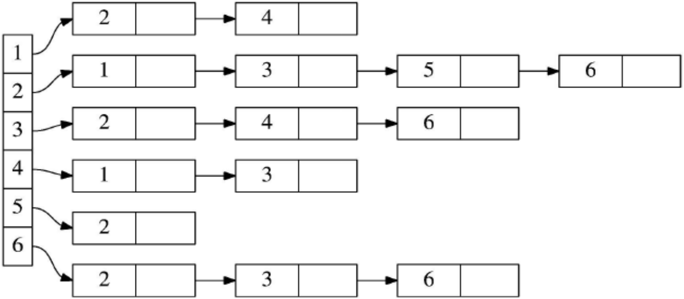
```
int a[46340][46340];
```

- Надо учесть, что часть памяти занята ОС и другими программами (предполагаем, что доступно 90% памяти (~ 7 Гб), тогда *n* = 43 347)
- Сколько поместится элементов типа unsigned char или uint8_t?

Списки смежных вершин

- **Списки смежных вершин** (adjacency list) – это массив $A[n]$, каждый элемент $A[i]$ которого содержит список узлов смежных с вершиной i

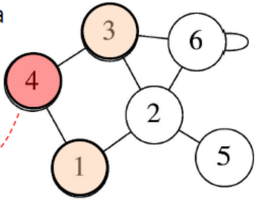




Эффективен для разреженных графов ($|E| \approx |V|$)

Массив смежных вершин. (CSR) представление

- Реализация списка смежных вершин на основе массивов $A[n+1]$ и $L[2m]$
- Список смежных вершин узла i: $L[A[i]]$, $L[A[i]+1]$, …, $L[A[i+1]-1]$



	1	2	3	4	5	6	7
A:	**1**	**3**	**7**	**10**	**12**	**13**	**15**

– индексы начала и конца списка смежных вершин в массиве L

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L:	**2**	**4**	**1**	**3**	**5**	**6**	**2**	**4**	**6**	**1**	**3**	**2**	**2**	**3**	**6**

Обход смежных вершин (CSR)

```
// Обход смежных вершин узла i
for (j = A[i]; j < A[i + 1]; j++) {
    v = L[j];
    // Обработать вершину v
}
```

$T = O(m)$

Как хранить большой разреженный граф в памяти?

Представление графа

- Матрица смежности
  - Невозможно сохранить граф с более чем 10^5 вершинами
- Списки смежности
  - Лучше, но требуются память для $4m$ целых чисел
- Массив смежных или сжатая разреженная строка (Compressed Sparse Row (**CSR**))
  - Представляет неориентированный граф $2m + n + O(1)$ целыми числами

n — количество вершин

m — количество ребер

Деревья

- Граф называется связным, если между любыми его двумя вершинами имеется путь.

Деревья

- Граф называется связным, если между любыми его двумя вершинами имеется путь.
- Дерево – это связный граф без циклов.

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).

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.

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;

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;

Деревья

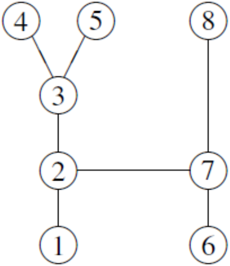
- Граф называется связным, если между любыми его двумя вершинами имеется путь.
- Дерево – это связный граф без циклов.
- Лес – это граф без циклов (или ациклический граф).
- Можно также сказать, что лес – это множество вершинно непересекающихся деревьев.

Теорема 1

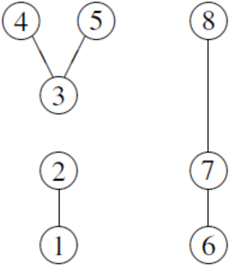
Для графа $G = (V, E)$ следующие условия эквивалентны:

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

Деревья минимальной стоимости

- Покрывающим (или остовным) деревом) графа $G = (V, E)$ называется
- такой его подграф $T = (V, E')(E' \subseteq E)$, который является деревом.

Деревья минимальной стоимости

- Покрывающим (или остовным) деревом) графа $G = (V, E)$ называется
- такой его подграф $T = (V, E')(E' \subseteq E)$, который является деревом.
- В задаче о минимальном остовном дереве

Деревья минимальной стоимости

- Покрывающим (или остовным) деревом) графа $G = (V, E)$ называется
- такой его подграф $T = (V, E')(E' \subseteq E)$, который является деревом.
- В задаче о минимальном остовном дереве
- в графе $G = (V, E)$, ребрам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$,

Деревья минимальной стоимости

- Покрывающим (или остовным) деревом) графа $G = (V, E)$ называется
- такой его подграф $T = (V, E')(E' \subseteq E)$, 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 $T = (V, E')$, у которого

Деревья минимальной стоимости

- Покрывающим (или остовным) деревом) графа $G = (V, E)$ называется
- такой его подграф $T = (V, E')(E' \subseteq E)$, 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 $T = (V, E')$, у которого
- сумма стоимостей его ребер $\sum_{(v,w)\in E'} c(v, w)$ минимальна.

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- Инициализация: выбрать произвольную вершину $s \in S$, положить

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ есть множество ребер минимального остовного дерева.
- Инициализация: выбрать произвольную вершину $s \in S$, положить
  - $S = \{s\}$, $parent(s) = s$;

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ 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 $v \in V \setminus \{s\}$;

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ 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 $(s, v) \in E(s, V)$.

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ 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 $S = V$,

Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ 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 $S = V$,
  - выбрать $v \in \arg\min_{w \in V \setminus S} d(w)$,

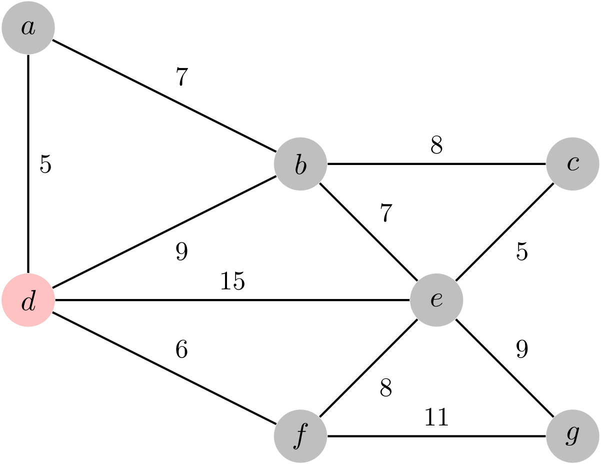
Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ 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 $S = V$,
  - выбрать $v \in \arg\min_{w \in V \setminus S} d(w)$,
  - положить $S := S \cup \{v\}$,

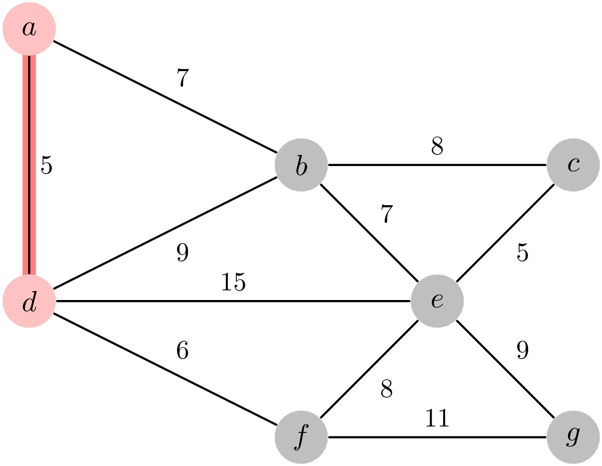
Алгоритм Прима

- Вход: граф $G = (V, E)$, функция стоимостей $c : E \to R$.
- Выход: функция $parent : V \to V$, что $E' = \{(parent(v), v) : v \in V, parent(v) \neq v\}$ 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 $S = V$,
  - выбрать $v \in \arg\min_{w \in V \setminus S} d(w)$,
  - 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 $d(w) = d(v) + c(v, w)$.

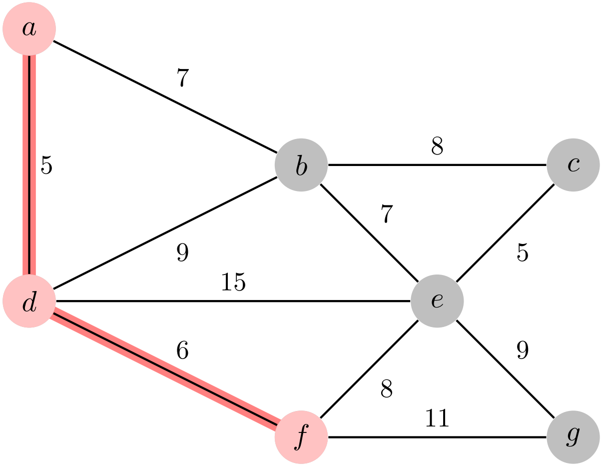
Алгоритм Прима. Пример



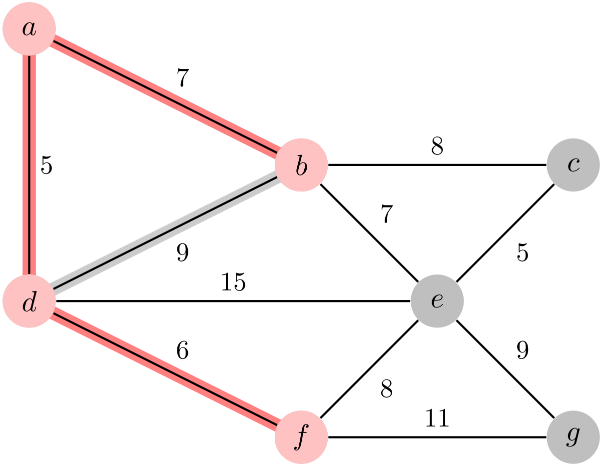
Алгоритм Прима. Пример



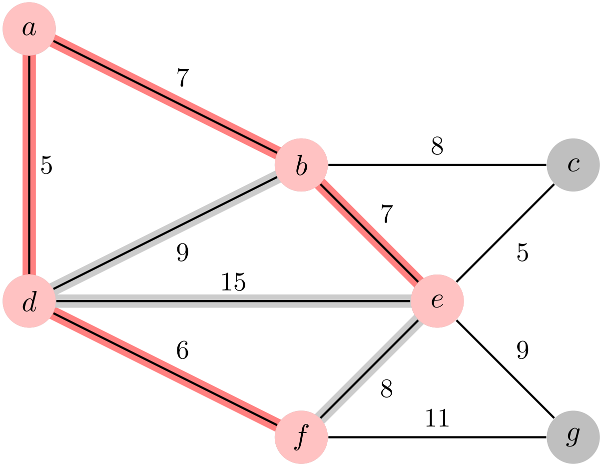
Алгоритм Прима. Пример



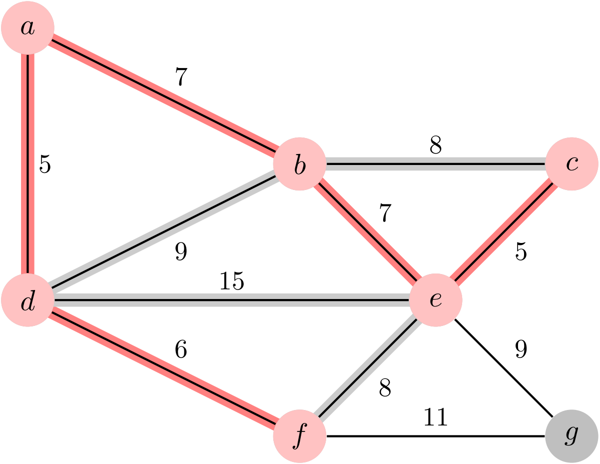
Алгоритм Прима. Пример



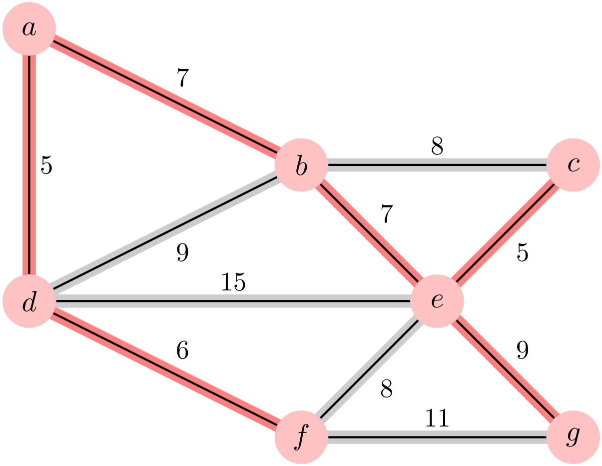
Алгоритм Прима. Пример



Алгоритм Прима. Пример



Алгоритм Прима. Пример



Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,

Кратчайшая связующая сеть дорог

- Пусть вершины графа $G = (V, E)$ представляют населенные пункты,
- а ребра – грунтовые дороги, соединяющие эти населенные пункты.

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.

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

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

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.

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.

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,

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.

Деревья Штейнера

- Пусть теперь вершины графа $G = (V, E)$ представляют населенные пункты $S \subseteq V$ и перекрестки дорог $V \setminus S$,

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.

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.

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

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

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.

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 $T = (V', E')$ минимальной стоимости, которое "покрывает" все населенные пункты, т. е. $S \subseteq V'$.

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 $T = (V', E')$ минимальной стоимости, которое “покрывает” все населенные пункты, т. е. $S \subseteq V'$.
- Такое дерево называется деревом Штейнера.

Ориентированны деревья

- Орграф $G = (V, E)$ называется ориентированным деревом (или ордеревом),

Ориентированны деревья

- Орграф $G = (V, E)$ называется ориентированным деревом (или ордеревом),
- если $|E| = |V| - 1$

Ориентированны деревья

- Орграф $G = (V, E)$ называется ориентированным деревом (или ордеревом),
- если $|E| = |V| - 1$
- и в каждую вершину входит не более одной дуги.

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.

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 ($parent(r) = r$).

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.

Поиск по графу

Поиск по графу

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют списком.

Очереди и стеки

- Множество, в котором задан порядок следования элементов, называют списком.
  - $\{x, y, z\}$ и $\{z, y, x\}$ – одно и то же множество,

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.

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:

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;

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.

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

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,

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;

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

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,

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 $s \in V$.

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.

Поиск по графу

- Задан орграф $G = (V, E)$ и вершина $s \in V$.
- Нужно найти все вершины, достижимые в G из s.
- Алгоритм:

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 $v \in V \setminus \{s\}$.

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 $Q = \emptyset$,

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 $Q = \emptyset$,
    - $v := pop(Q)$;

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 $Q = \emptyset$,
    - $v := pop(Q)$;
    - Для всех $(v, w) \in E(v, V)$, если $parent(w) = nil$, полагаем $parent(w) := v$ и выполняем $push(Q, w)$.

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 $Q = \emptyset$,
    - $v := pop(Q)$;
    - Для всех $(v, w) \in E(v, V)$, если $parent(w) = nil$, полагаем $parent(w) := v$ и выполняем $push(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают дерево поиска.

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 $Q = \emptyset$,
        - $v := pop(Q)$;
        - Для всех $(v, w) \in E(v, V)$, если $parent(w) = nil$, полагаем $parent(w) := v$ и выполняем $push(Q, w)$.
- После завершения работы алгоритма указатели $parent(v)$ $(v \in V)$ задают дерево поиска.

Поиск по графу называется

- поиском в ширину, если Q есть очередь;

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 $Q = \emptyset$,
    - $v := pop(Q)$;
    - Для всех $(v, w) \in E(v, V)$, если $parent(w) = nil$, полагаем $parent(w) := v$ и выполняем $push(Q, w)$.
- 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 в глубину

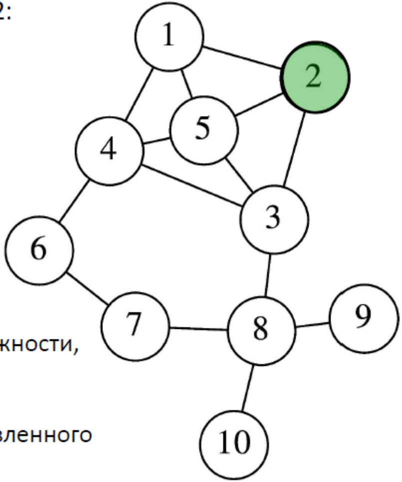
- **Поиск в глубину** (depth-first search – DFS) – процедура посещения всех вершин графа начиная с заданного узла *v*
- Сперва посещаем (обрабатываем) все самые “глубокие” вершины

```
function DFS(v)
    visited[v] = true
    // Обрабатываем данные вершины v
    for each u in Adj(v) do    // Перебор смежных вершин
        if visited[u] = false then
            DFS(u)  // Рекурсивно обрабатываем вершину u
        end if
    end for
end function
```

Поиск в глубину

- Обход в глубину с вершины 2:
  DFS(2)

```
for each u in Adj(2) do
    ...
end for
```

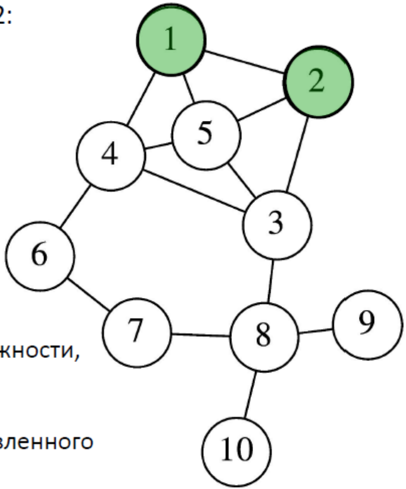


- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2:
  DFS(2)

```
for each u in Adj(2) do
    ...
end for
```

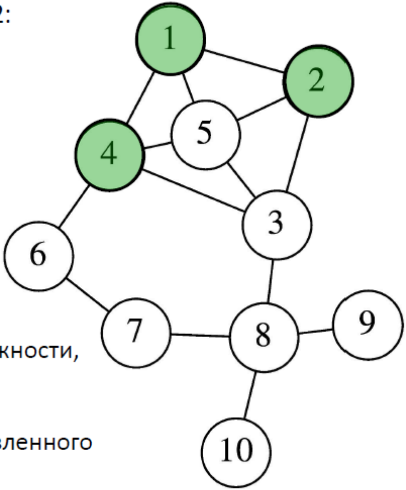


- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2:
  DFS(2)

```
for each u in Adj(2) do
    ...
end for
```

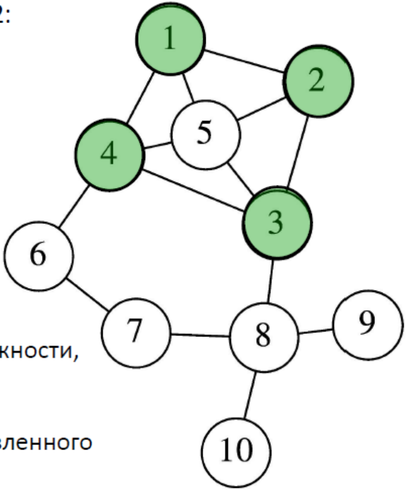


- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2:
  DFS(2)

```
for each u in Adj(2) do
    ...
end for
```



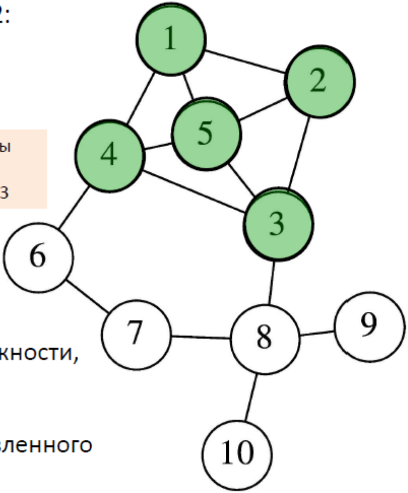
- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2: DFS(2)

```
for each u in Adj(2) do
    ...
end for
```

- Все смежные вершины узла 5 посещены
- Возвращаемся к узлу 3

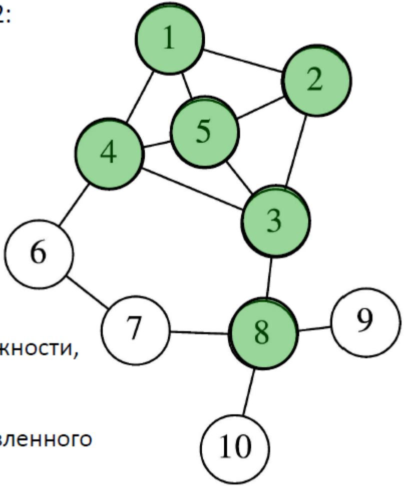


- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2:
  DFS(2)

```
for each u in Adj(2) do
    ...
end for
```



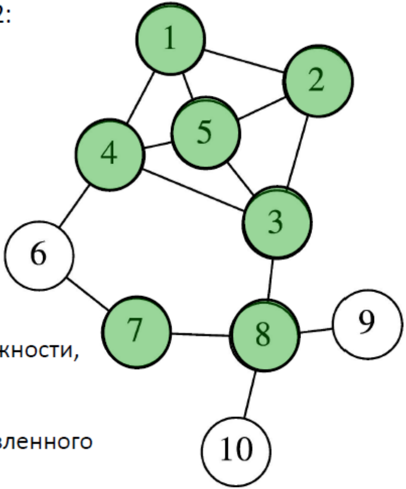
- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$

- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2:
  DFS(2)

```
for each u in Adj(2) do
    ...
end for
```



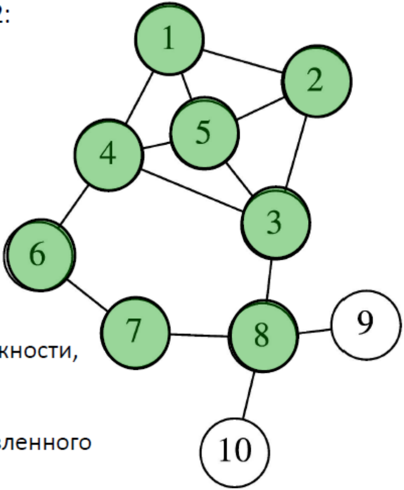
- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2: DFS(2)

```
for each u in Adj(2) do
    ...
end for
```

- Все смежные вершины узла 6 посещены
- Возвращаемся к узлу 7, затем к 8

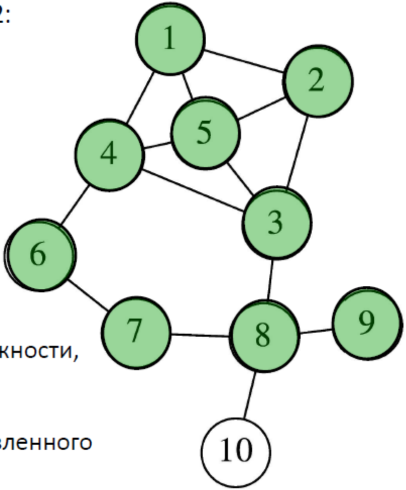


- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2:
  DFS(2)

```
for each u in Adj(2) do
    ...
end for
```

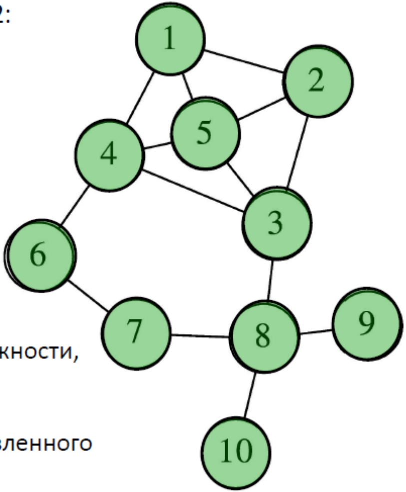


- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в глубину

- Обход в глубину с вершины 2:
DFS(2)

```
for each u in Adj(2) do
    ...
end for
```



- Обход в глубину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в глубину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

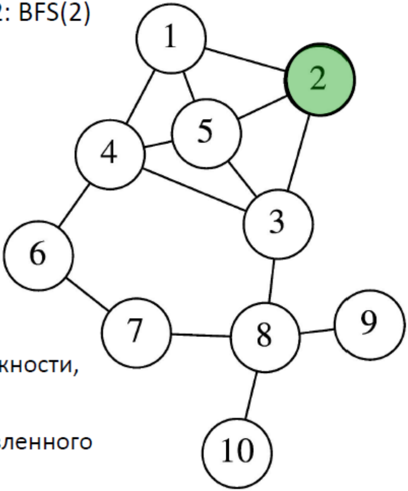
- **Поиск в ширину** (breadth-first search – BFS, обход в ширину) – процедура посещения всех вершин графа начиная с заданного узла *v*
- Сперва посещаем (обрабатываем) свои дочерние вершины

Поиск в ширину

```
function BFS(v)
    visited[v] = true
    // Обрабатываем вершину v
    QueueEnqueue(v)        // Помещаем v в очередь вершин
    while QueueSize() > 0 do
        u = QueueDequeue()           // Извлекаем вершину
        for each x in Adj(u) do
            if visited[x] = false then
                QueueEnqueue(x)
                visited[x] = true
                // Обрабатываем узел x
            end if
        end for
    end while
end function
```

Поиск в ширину

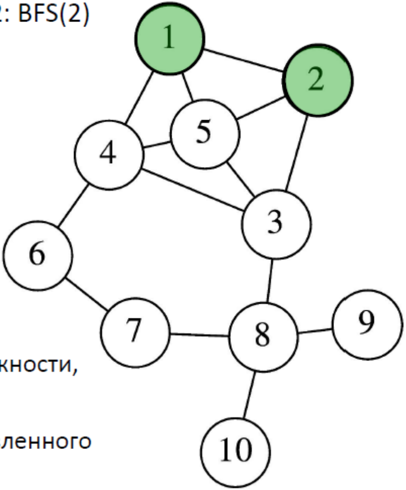
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **2**
- В очереди:
  1, 3, 5



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

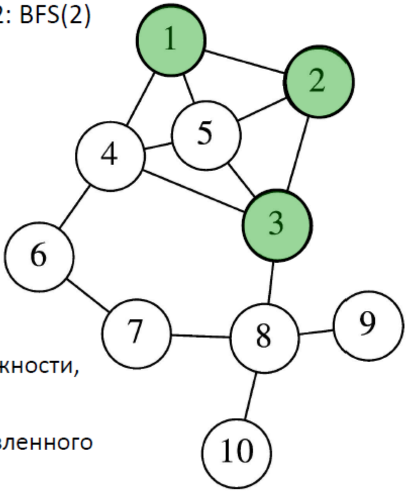
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **1**
- В очереди:
  3, 5, **4**



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

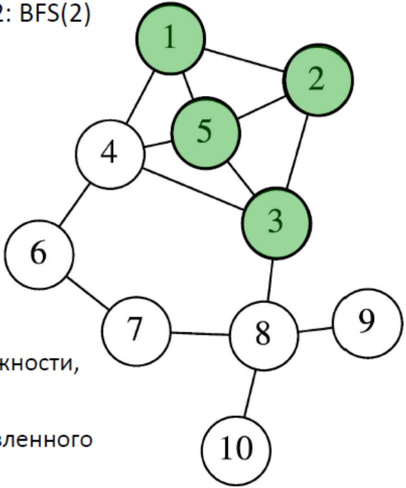
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **3**
- В очереди:
  5, 4, **8**



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

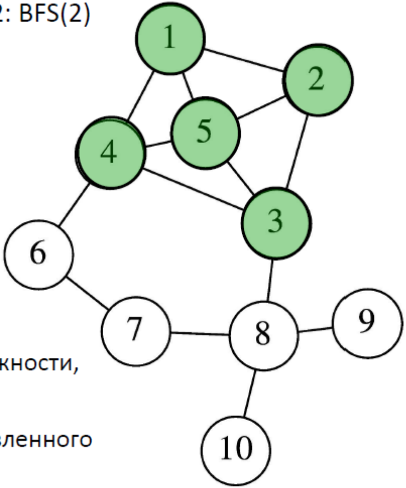
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **5**
- В очереди:
  4, 8



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

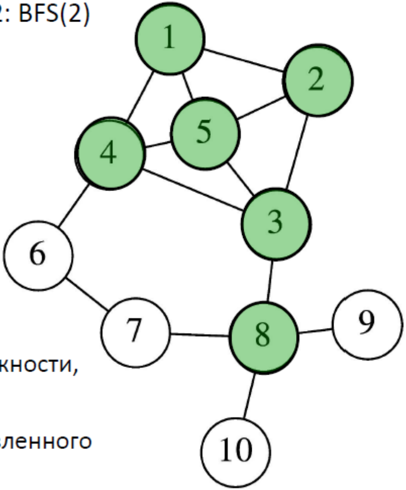
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **4**
- В очереди:
  8, **6**



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

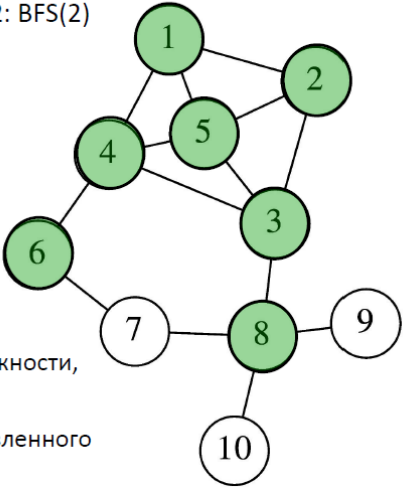
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **8**
- В очереди:
  6, **7, 9, 10**



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

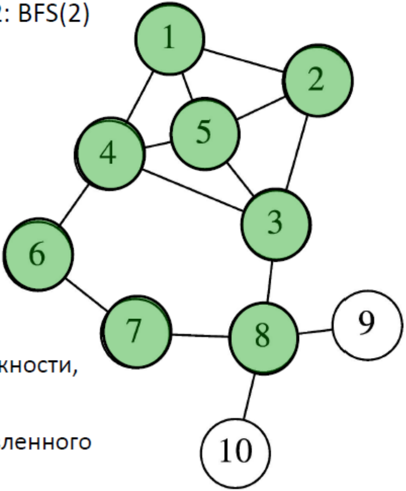
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **6**
- В очереди:
  7, 9, 10



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

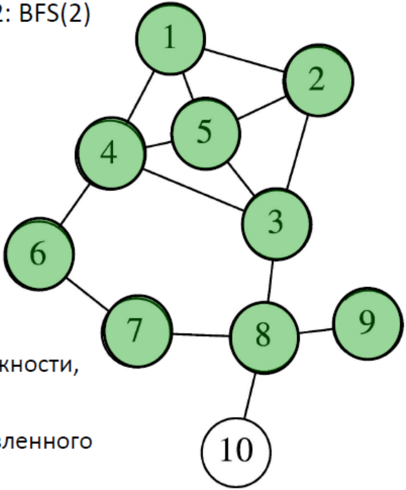
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **7**
- В очереди:
  9, 10



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

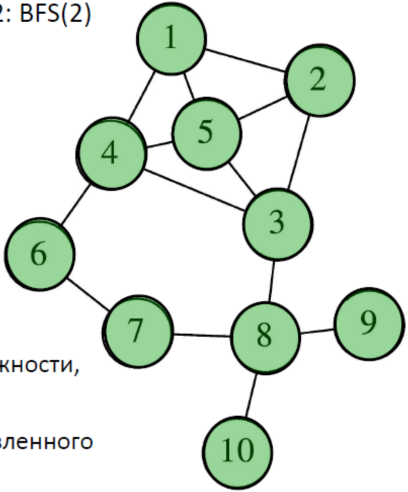
- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **9**
- В очереди:
  10



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

Поиск в ширину

- Обход в ширину с вершины 2: BFS(2)
- Извлекли из очереди: **10**
- В очереди:



- Обход в ширину графа, представленного матрицей смежности, имеет трудоемкость $O(|V|^2)$
- Обход в ширину графа, представленного списком смежности, имеет трудоемкость $O(|V| + |E|)$

План лекции

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

Эйлеровы циклы

Определение

Цикл в мультиграфе (не обязательно простой), который проходит по каждому ребру ровно один раз называется эйлеровым.

Эйлеровы циклы

Определение

Цикл в мультиграфе (не обязательно простой), который проходит по каждому ребру ровно один раз называется эйлеровым.

Определение

Графы (мультиграфы), содержащие эйлеровы циклы, называются эйлеровыми графами (мультиграфами).

Характеризация эйлеровых графов

Теорема (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он

Характеризация эйлеровых графов

Теорема (Эйлера)

Мультиграф G эйлеров тогда и только тогда, когда он связен

Характеризация эйлеровых графов

Теорема (Эйлера)

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

Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.

Гамильтоновы графы

- Рассмотрим граф $G = (V, E)$.
- Простой цикл, содержащий все $n = |V|$ вершин графа, называется гамильтоновым.

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.

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.

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.

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,
- нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,
- нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- наименьшего веса $c(\Gamma) = \sum_{i=1}^{n} c(v_{i-1}, v_i)$.

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,
- нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- наименьшего веса $c(\Gamma) = \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- Задача коммивояжера – это задача о гамильтоновом цикле наименьшего веса в полном графе.

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,
- нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- наименьшего веса $c(\Gamma) = \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- Задача коммивояжера – это задача о гамильтоновом цикле наименьшего веса в полном графе.
- Вершины графа представляют некоторые города, а вес $c(v, w)$ – это расстояние между городами.

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,
- нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- наименьшего веса $c(\Gamma) = \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- Задача коммивояжера – это задача о гамильтоновом цикле наименьшего веса в полном графе.
- Вершины графа представляют некоторые города, а вес $c(v, w)$ – это расстояние между городами.
- Коммивояжер, начиная из города, в котором он проживает,

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,
- нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \ldots, v_n = v_0)$
- наименьшего веса $c(\Gamma) = \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- Задача коммивояжера – это задача о гамильтоновом цикле наименьшего веса в полном графе.
- Вершины графа представляют некоторые города, а вес $c(v, w)$ – это расстояние 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,

Задача о гамильтоновом цикле наименьшего веса

- Каждому ребру $(v, w) \in E$ графа $G = (V, E)$ приписан вес $c(v, w)$,
- нужно найти гамильтоном цикл $\Gamma = (v_0, v_1, \dots, v_n = v_0)$
- наименьшего веса $c(\Gamma) = \sum_{i=1}^{n} c(v_{i-1}, v_i)$.
- Задача коммивояжера – это задача о гамильтоновом цикле наименьшего веса в полном графе.
- Вершины графа представляют некоторые города, а вес $c(v, w)$ – это расстояние 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 максимальной клике

- Граф $H = (V', E')$ называется подграфом графа $G = (V, E)$, если $V' \subseteq V$ и $E' \subseteq E$.

Задача о максимальной клике

- Граф $H = (V', E')$ называется подграфом графа $G = (V, E)$, если $V' \subseteq V$ и $E' \subseteq E$.
- Максимальный (по включению) полный подграф графа G называется кликой.

Задача о максимальной клике

- Граф $H = (V', E')$ называется подграфом графа $G = (V, E)$, если $V' \subseteq V$ и $E' \subseteq E$.
- 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.

Задача о максимальной клике

- Граф $H = (V', E')$ называется подграфом графа $G = (V, E)$, если $V' \subseteq V$ и $E' \subseteq E$.
- 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.

Задача о максимальной клике

- Граф $H = (V', E')$ называется подграфом графа $G = (V, E)$, если $V' \subseteq V$ и $E' \subseteq E$.
- 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.

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

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,

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,

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.

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,

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.

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:

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,

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.

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.

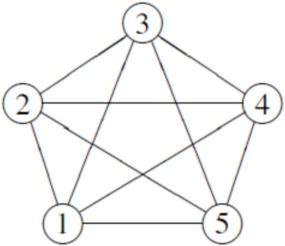
Укладка графа на плоскости

- 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.

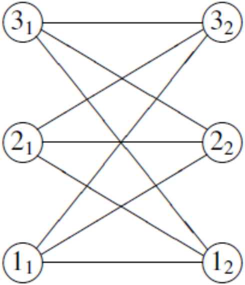
Укладка графа на плоскости

- 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.

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.

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.

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 (Куратовского)

Граф G планарен тогда и только тогда, когда

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 (Куратовского)

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