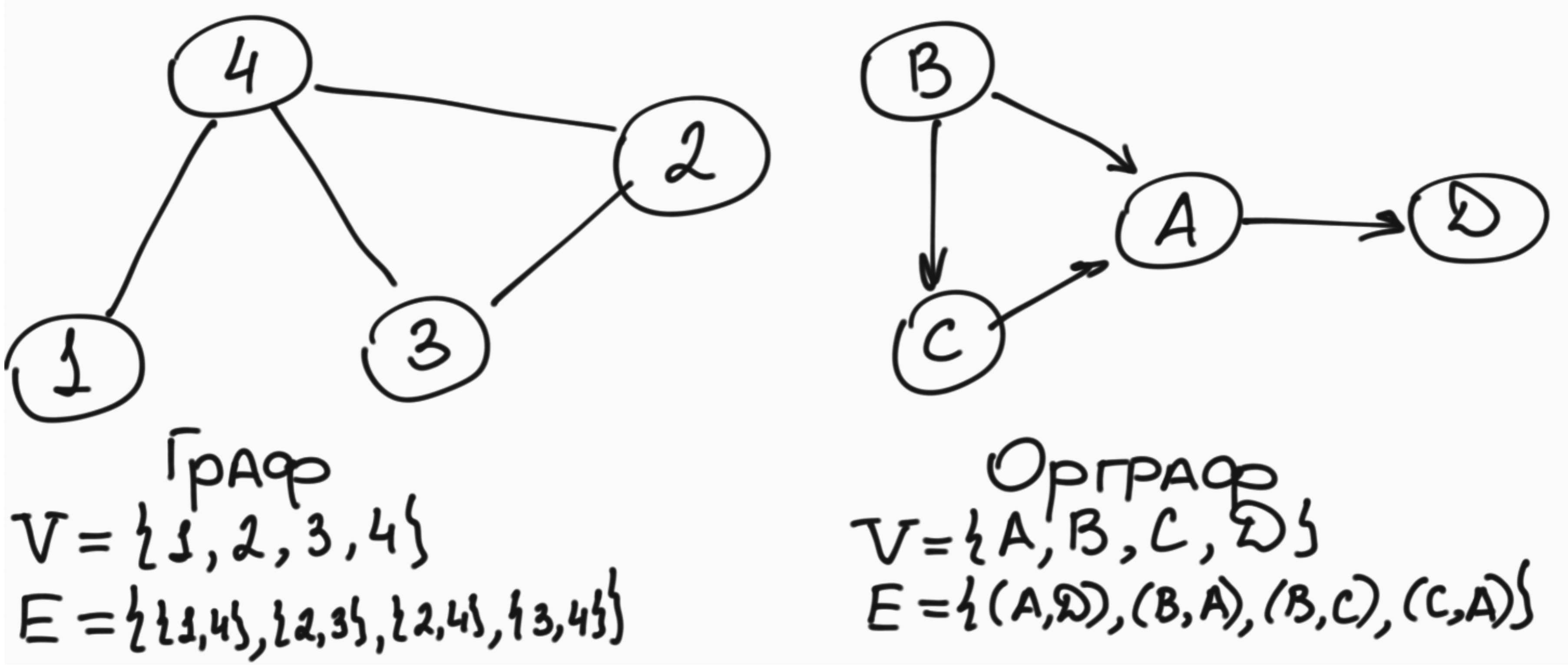
Основы теории графов

Основы теории графов

Граф - пара множеств (V, E), где V - множество вершин, $E \subset V \times V$ - множество ребер.

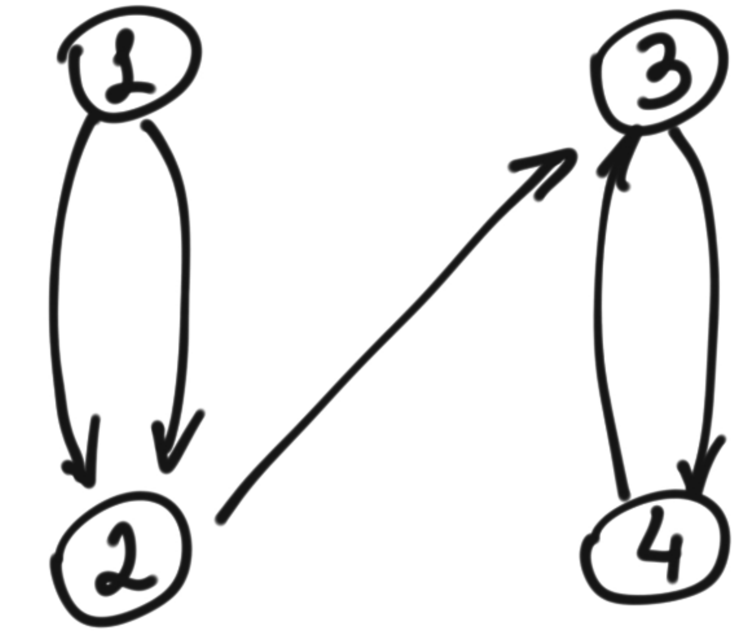
Если пары вершин в ребрах упорядочены, то такой граф называется *ориентированным* (орграф).



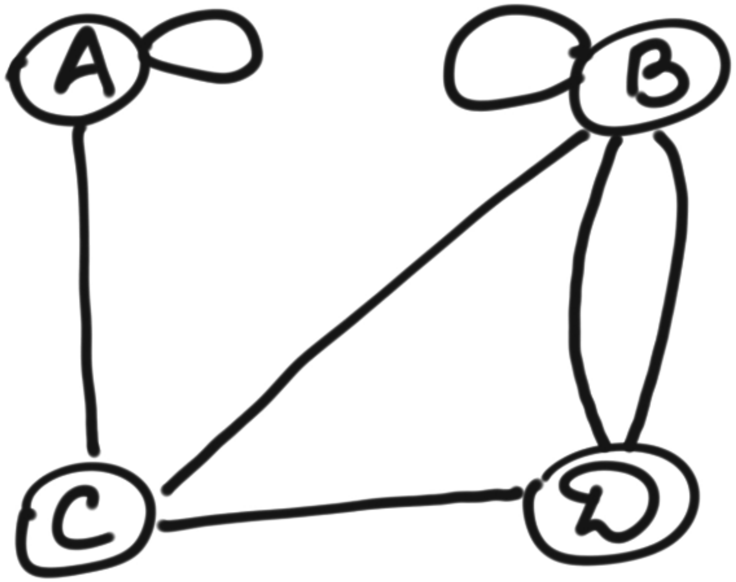
Основы теории графов

Если множество ребер - мультимножество, то граф называется *мультиграфом*.

Если допускаются ребра из вершины в нее саму, то граф - *псевдограф*.



Мультиорграф



Мультипсевдограф

Основы теории графов

Степенью вершины v в неографе называется число $deg(v) := |\{e \in E | v \in e\}|$, то есть число инцидентных ребер.

Полустепенью исхода вершины v в орграфе называется число $deg_+(v) := |\{(v, u) | (v, u) \in E\}|$, то есть число исходящих ребер.

Полустепенью захода вершины v в орграфе называется число $deg_-(v) := |\{(u, v) | (u, v) \in E\}|$, то есть число входящих ребер.

Утверждение.

Для неориентированных графов $\sum\limits_{v \in V} deg(v) = 2|E|$

Для ориентированных графов $\sum\limits_{v \in V} deg_+(v) = \sum\limits_{v \in V} deg_-(v) = |E|$

Основы теории графов

Путем в графе называется последовательность вершин и ребер, причем каждое ребро соединяет соседние вершины в последовательности.

Простой путь - путь, в котором нет повторяющихся вершин.

Цикл - замкнутый путь (начало совпадает с концом) без повторяющихся ребер.

Представление графов в памяти

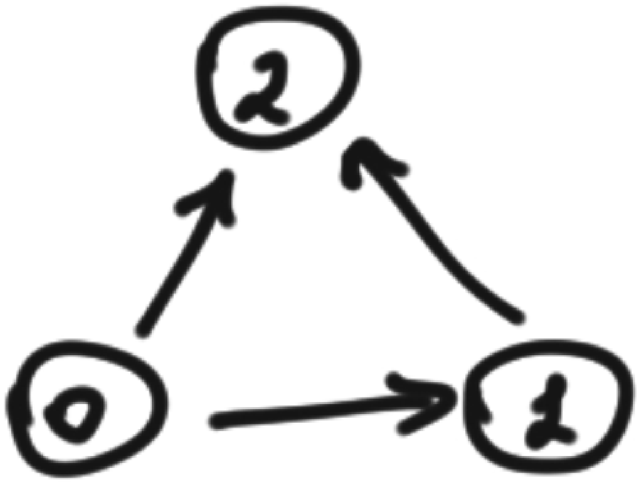
Представление вершин

Вершины, как правило, будем задавать последовательностью чисел $0, 1, ..., |V| - 1$

Далее, для краткости вместо $|V|$ и $|E|$ будем использовать V и E, если из контекста понятно, что имеется в виду число.

Список ребер

Ребра можно представлять в виде списка/массива пар вершин.

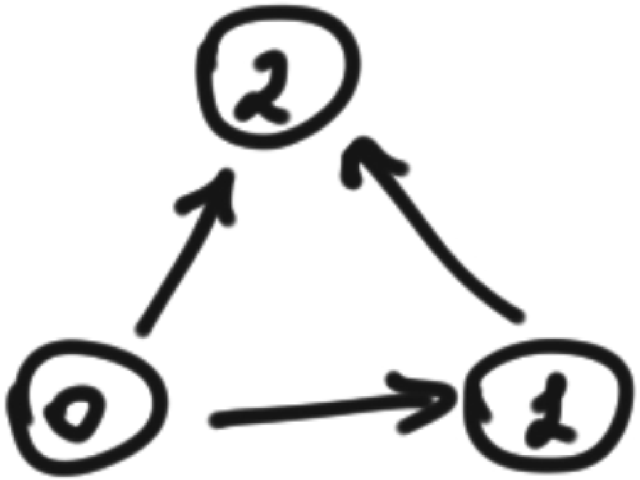


[(0,1), (1,2), (0,2)]

- Это просто.
- Память - ?
- Обход всех ребер - ?
- Поиск ребра - ?
- Получение соседей вершины - ?

Список ребер

Ребра можно представлять в виде списка/массива пар вершин.

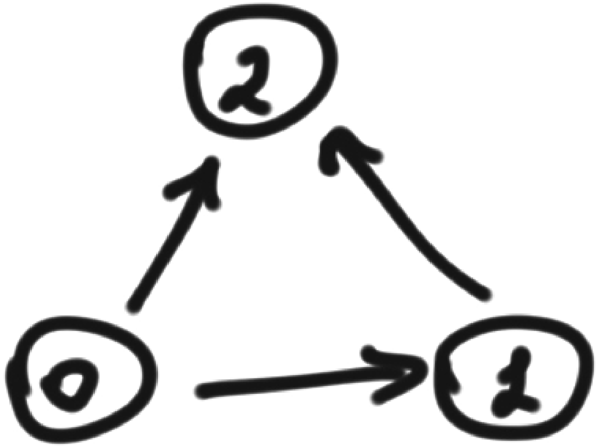


[(0,1),(1,2),(0,2)]

- Это просто.
- Память - $\Theta(E)$
- Обход всех ребер - $\Theta(E)$
- Поиск ребра - $O(E)$
- Получение соседей вершины - $\Theta(E)$

Сортированный список ребер

Предварительно список ребер можно отсортировать по началу (и в каждой группе дополнительно по концу).

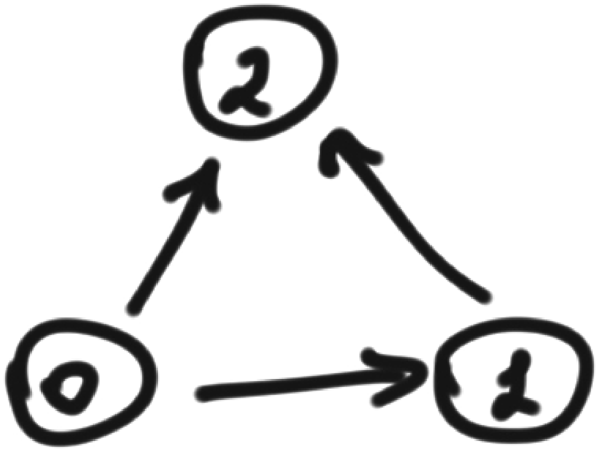


[(0,1), (0,2), (1,2)]

- Требует времени, неэффективная вставка/удаление ребра
- Память - $\Theta(E)$
- Обход всех ребер - $\Theta(E)$
- Поиск ребра - ?
- Получение соседей вершины - ?

Сортированный список ребер

Предварительно список ребер можно отсортировать по началу (и в каждой группе дополнительно по концу).

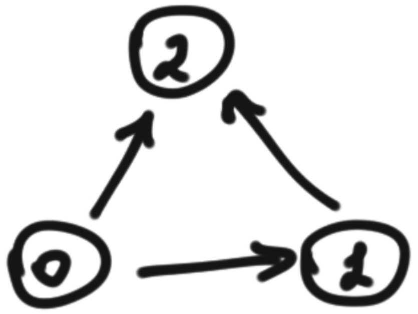


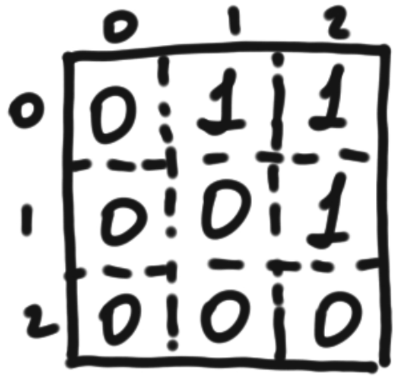
[(0,1), (0,2), (1,2)]

- Требует времени, неэффективная вставка/удаление ребра
- Память - $\Theta(E)$
- Обход всех ребер - $\Theta(E)$
- Поиск ребра - $O(\log E)$
- Получение соседей вершины - $O(\log E + deg_{+}(v))$ в случае орграфа.

Матрица смежности

Заведем таблицу $V \times V$, в ячейке (i, j) будем хранить метку наличие/отсутствие ребра (либо количество ребер) из i в j.

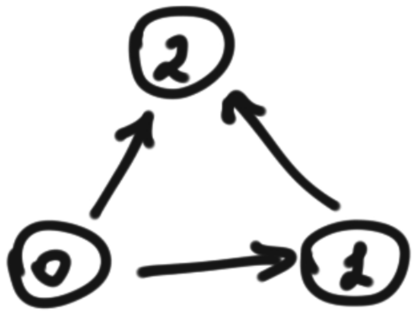


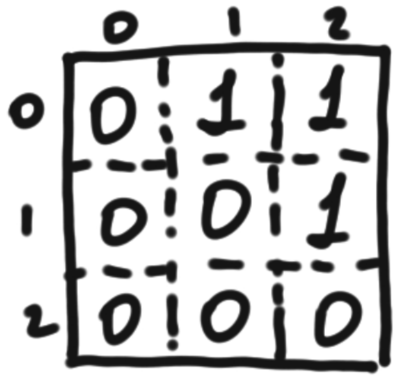


- Неэффективна для разреженных графов ($E \sim V$)
- Память - $\Theta(V^2)$
- Обход всех ребер - ?
- Поиск ребра - ?
- Получение соседей вершины - ?

Матрица смежности

Заведем таблицу $V \times V$, в ячейке (i, j) будем хранить метку наличие/отсутствие ребра (либо количество ребер) из i в j.





- Неэффективна для разреженных графов ($E \sim V$)
- Память - $\Theta(V^2)$
- Обход всех ребер - $\Theta(V^2)$
- Поиск ребра - $O(1)$
- Получение соседей вершины - $\Theta(V)$

Списки смежности

Заведем массив/список из V массивов/списков. В i-м массиве храним соседей вершины i.

- Память - $\Theta(V + E)$
- Обход всех ребер - ?
- Поиск ребра - ?
- Получение соседей вершины - ?

Списки смежности

Заведем массив/список из V массивов/списков. В i-м массиве храним соседей вершины i.

- Память - $\Theta(V + E)$
- Обход всех ребер - $\Theta(V + E)$
- Поиск ребра - $O(deg_+(v))$
- Получение соседей вершины - $O(1)$ (если не требуется копирование)

Списки смежности

Заведем массив/список из V массивов/списков. В i-м массиве храним соседей вершины i.

Замечание: вместо массивов/списков соседей можно хранить BST или хеш-таблицу, тогда время поиска ребра будет асимптотически быстрее, но на хранение будет уходить больше памяти.

Обход графа в ширину

Breadth-First Search (BFS)

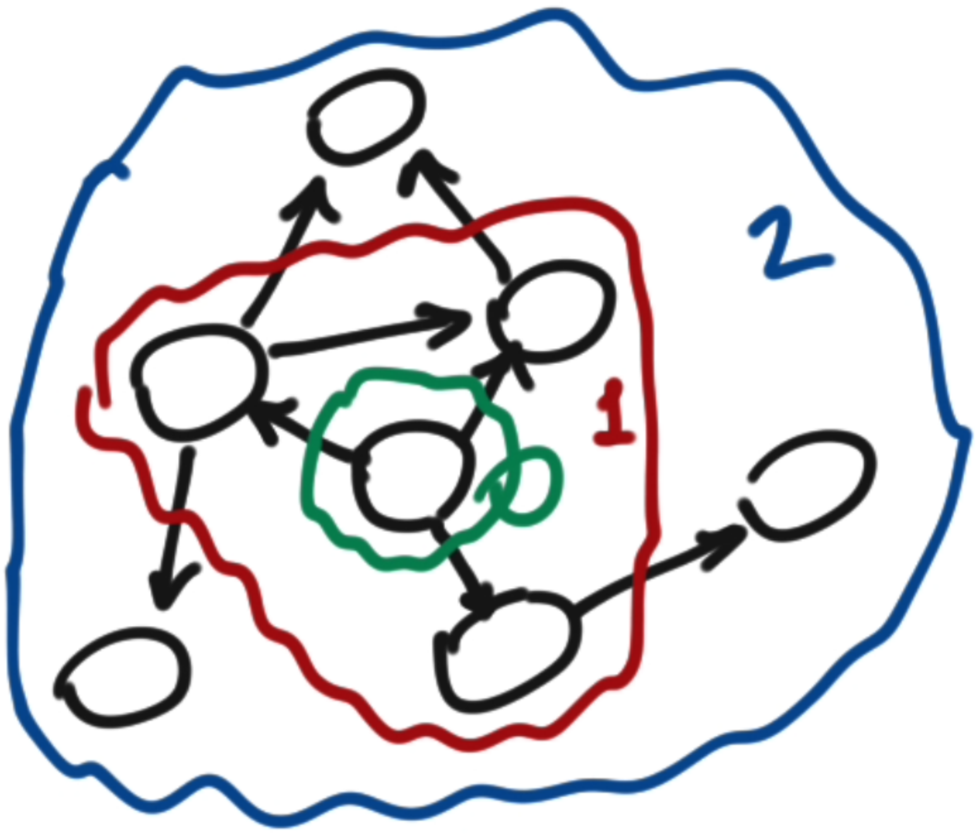
Обход графа в ширину (BFS)

Обход графа - процесс посещения всех вершин и ребер графа.

Как правило, обходы осуществляются в определенной последовательности, с учетом структуры графа, с целью выявить некоторые его свойства.

Обход графа в ширину (BFS)

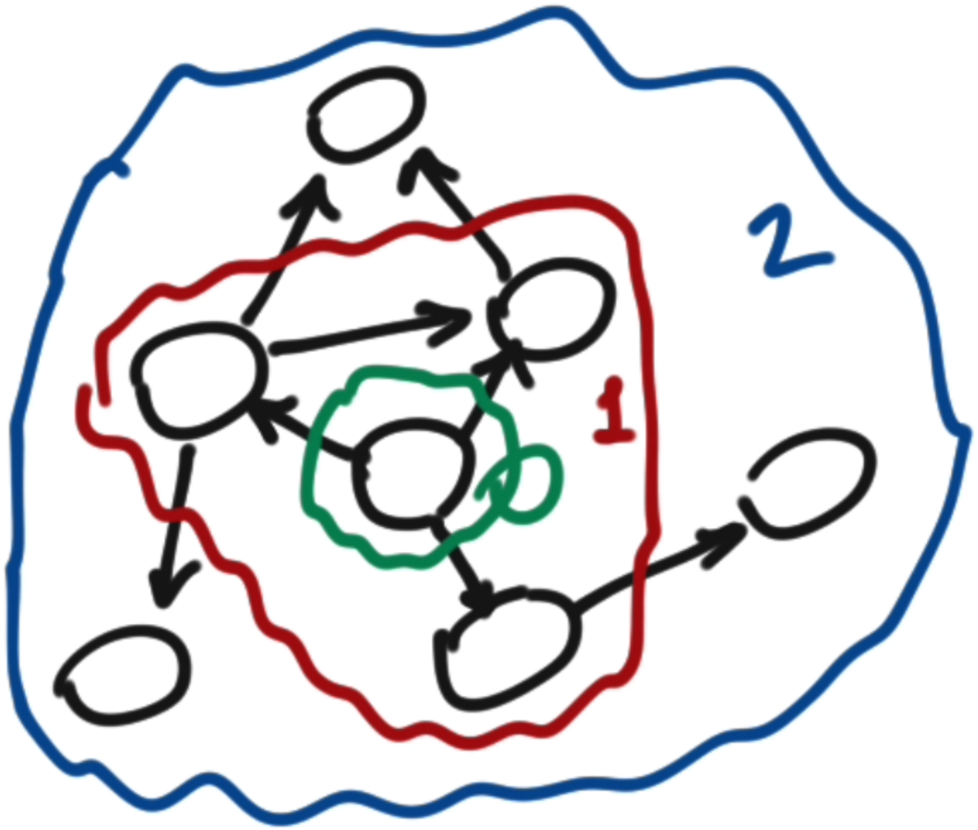
Метод Винни-Пуха: пойдем сначала к соседям, затем к соседям соседей и т.д.



Какие свойства может извлечь такой обход?

Обход графа в ширину (BFS)

Метод Винни-Пуха: пойдем сначала к соседям, затем к соседям соседей и т.д.



- Вершины достижимые из данной
- Кратчайшие пути из данной вершины

Поиск кратчайших путей с помощью BFS

```
def BFS(G, s):  # G - граф, s - стартовая вершина
  dist = [inf, inf, ..., inf]
  parent = [None, None, ..., None]  # для восстановления пути
  queue = {s}  # очередь обхода
  dist[s] = 0
  while not queue.empty():
    v = queue.pop()
    for u in G.neighbors(v):
      if dist[u] == inf:
        dist[u] = dist[v] + 1
        parent[u] = v
        queue.push(u)
  return dist, parent
```

Время работы - ?

Поиск кратчайших путей с помощью BFS

```
def BFS(G, s):  # G - граф, s - стартовая вершина
  dist = [inf, inf, ..., inf]
  parent = [None, None, ..., None]  # для восстановления пути
  queue = {s}  # очередь обхода
  dist[s] = 0
  while not queue.empty():
    v = queue.pop()
    for u in G.neighbors(v):
      if dist[u] == inf:
        dist[u] = dist[v] + 1
        parent[u] = v
        queue.push(u)
  return dist, parent
```

Время работы - $O(V+E)$, если списки смежности, и $O(V^2)$, если матрица смежности.

Корректность

Докажем, что массив $dist$ действительно хранит кратчайшие расстояния из s.

Лемма 1. *Вершины в очереди расположены по неубыванию* $dist$ *и их расстояния различаются максимум на 1*.

Корректность

Лемма 1. *Вершины в очереди расположены по неубыванию* $dist$ *и их расстояния различаются максимум на 1*.

Доказательство.
По индукции. База: для очереди из одной начальной вершины это верно.
Переход: пусть это верно на произвольной итерации k. Тогда в начале очереди расположены вершины со значением d, а в конце со значением $d + 1$.
Достали вершину и добавили какое-то количество вершин со значением $d + 1$.
То есть в начале по-прежнему идет какое-то количество (возможно, 0) вершин со значением d, а затем - $d + 1$. ∎

Корректность

Лемма 2. $\forall v \in V : dist[v] \geq \rho(s, v)$, *где* $\rho(s, v)$ *- расстояние от* s *до* v

Корректность

Лемма 2. $\forall v \in V : dist[v] \geq \rho(s, v)$, *где* $\rho(s, v)$ *- расстояние от* s *до* v

Доказательство.

По индукции. База: $dist[s] == 0 \geq \rho(s, s) == 0$.
Переход: пусть это верно после первых k итераций. На $k + 1$ шаге достаем вершину v ($dist[v] \geq \rho(s, v)$) и добавляем ее соседей u: $dist[u] = dist[v] + 1 \geq \rho(s, v) + 1 \geq \rho(s, u)$. Последнее неравенство следует из того, что путь из s в u, проходящий через v не короче кратчайшего пути из s в u. $\blacksquare$.

Корректность

Теорема (о корректности BFS).

После работы BFS $\forall v \in V : dist[v] == \rho(s, v)$

Корректность

Теорема (о корректности BFS).

После работы BFS $\forall v \in V : dist[v] == \rho(s, v)$

Доказательство.

От противного. Пусть $\exists x \in V : dist[x] > \rho(s, x)$.

Положим $v := \underset{x : dist[x] > \rho(s,x)}{\arg\min} \rho(s, x)$.

Пусть $u := parent[v]$, а p - истинный предок v на кратчайшем пути из s.

$$\begin{cases} dist[v] = dist[u] + 1 \\ \rho(s, v) = \rho(s, p) + 1 \Rightarrow dist[u] > \rho(s, p) = dist[p] \Rightarrow \text{вершину } p \text{ достали} \\ dist[v] > \rho(s, v) \end{cases}$$

раньше вершины u, а значит вершину v обработали раньше u. Противоречие. ■

BFS для взвешенных графов

0-1 граф

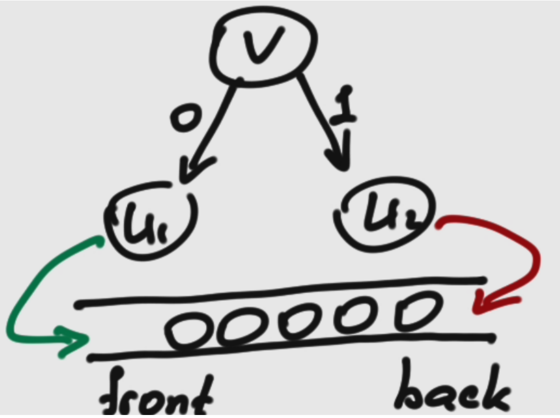
Пусть каждое ребро имеет вес 0 или 1, то есть какие-то ребра не учитываются при подсчете кратчайшего расстояния.

Что делать?

0-1 граф

Пусть каждое ребро имеет вес 0 или 1, то есть какие-то ребра не учитываются при подсчете кратчайшего расстояния.

- Тогда вершины, к которым ведут нулевые ребра кладем в начало очереди, а остальные - в конец.
- Легко заметить, что все утверждения лемм и теоремы останутся верными.
- + в алгоритме заменить проверку $dist[u] == inf$ на $dist[v] + w(v, u) < dist[u]$ (почему?)



0-k граф: 1 способ

Пусть каждое ребро имеет вес из множества $\{0, 1, ..., k\}$.

Что делать?

0-k граф: 1 способ

Пусть каждое ребро имеет вес из множества $\{0, 1, ..., k\}$.

- Разобьем ребро веса $w > 1$ на w ребер, добавив фиктивные вершины. Свели задачу к предыдущей.
- Новое число вершин и ребер $V' \leq (k-1)E + V$, $E' \leq kE$.
- Сложность $O(kE + V)$.

0-k граф: 2 способ

Пусть каждое ребро имеет вес из множества $\{0, 1, ..., k\}$.

- Заведем $k(V-1)$ очередей. В очередь i будем складывать вершины с $dist[v] == i$. (Вопрос: почему очередей именно столько?)
- Обрабатывать очереди будем... по очереди.
- Таким образом, упорядочим все вершины в порядке возрастания $dist$.
- Суммарно добавим не более $E + V$ вершин + обход очередей за $O(kV)$
- Итоговое время работы $O(kV + E)$, память $O(kV + E)$

0-k граф: 2 способ

Пусть каждое ребро имеет вес из множества $\{0, 1, ..., k\}$.

- Заметим, что в каждый момент времени "работает" только $k + 1$ очередь, так как заполняются только очереди максимум на k шагов вперед.
- Следовательно, достаточно хранить $k + 1$ очередь и в очередь i добавлять вершины с $dist[v] \% (k + 1) == i$.
- Итоговое время работы $O(kV + E)$, память $O(k + E)$