# Введение в Python

## Графы. Обходы графов, расстояние между вершинами.

Алексей Андреевич Сорокин

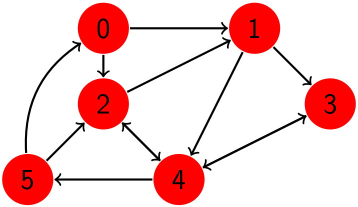
спецкурс, ОТИПЛ МГУ,
осенний семестр 2017–2018 учебного года, 21 ноября

# Графы

- Формально, граф это кортеж $G = \langle V, E \rangle$, где $V$ — конечное множество вершин, $E \subset V \subset V$ — множество рёбер.
- По умолчанию граф *ориентированный* — из того, что $\langle u, v \rangle \in E$ не следует, что $\langle v, u \rangle \in E$ (то есть рёбра — упорядоченные пары вершин).
- Если условия $\langle u, v \rangle \in E$, $\langle v, u \rangle \in E$ равносильны (рёбра — неупорядоченные пары), то граф *неориентированный*.

# Графы

Графы принято изображать с помощью диаграмм:




Представление в памяти: матрица смежности

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 1 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 |

# Графы

- Представление графа в памяти: матрица смежности

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 1 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 |

# Графы

- Представление графа в памяти: матрица смежности

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 1 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 |

- Другое представление: списки смежности

$$0 : [1, 2], 1 : [3, 4], 2 : [1, 4], 3 : [4], 4 : [1, 3, 5], 5 : [0, 2]$$

# Графы

- Представление графа в памяти: матрица смежности

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 1 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 |

- Другое представление: списки смежности

$$0 : [1, 2], 1 : [3, 4], 2 : [1, 4], 3 : [4], 4 : [1, 3, 5], 5 : [0, 2]$$

- Списки используются чаще, т. к. занимают меньше места.
- Матрицы используются реже, хотя удобны для формальных рассуждений.

# Обходы графов

- Во многих приложениях нужно уметь выписывать все вершины графа по одному разу, начиная с некоторой.

# Обходы графов

- Во многих приложениях нужно уметь выписывать все вершины графа по одному разу, начиная с некоторой.
- Это делается с помощью обходов в глубину или в ширину.

# Обходы графов

- Во многих приложениях нужно уметь выписывать все вершины графа по одному разу, начиная с некоторой.
- Это делается с помощью обходов в глубину или в ширину.
- Основная идея обходов:
    - На каждом шаге рассмотреть очередную необработанную вершину.

# Обходы графов

- Во многих приложениях нужно уметь выписывать все вершины графа по одному разу, начиная с некоторой.
- Это делается с помощью обходов в глубину или в ширину.
- Основная идея обходов:
  - На каждом шаге рассмотреть очередную необработанную вершину.
  - Пометить эту вершину некоторым образом.

# Обходы графов

- Во многих приложениях нужно уметь выписывать все вершины графа по одному разу, начиная с некоторой.
- Это делается с помощью обходов в глубину или в ширину.
- Основная идея обходов:
  - На каждом шаге рассмотреть очередную необработанную вершину.
  - Пометить эту вершину некоторым образом.
  - До/после обработки данной вершины осуществить обход из всех её нерассмотренных соседей.

# Обходы графов

- Во многих приложениях нужно уметь выписывать все вершины графа по одному разу, начиная с некоторой.
- Это делается с помощью обходов в глубину или в ширину.
- Основная идея обходов:
  - На каждом шаге рассмотреть очередную необработанную вершину.
  - Пометить эту вершину некоторым образом.
  - До/после обработки данной вершины осуществить обход из всех её нерассмотренных соседей.
- Для упорядочивания вершин используется очередь (обход в ширину) или стек (обход в глубину).

# Обход в ширину: описание

Обход в ширину осуществляется с помощью очереди.
Алгоритм обхода.

- Поместить в очередь стартовую вершину, пометить её как просмотренную.

# Обход в ширину: описание

Обход в ширину осуществляется с помощью очереди.

Алгоритм обхода.

- Поместить в очередь стартовую вершину, пометить её как просмотренную.
- Пока очередь непуста:
  - Извлечь из очереди первую вершину.
  - Добавить в очередь всех её непросмотренных соседей, помечая их как просмотренных.

# Обход в ширину: описание

Обход в ширину осуществляется с помощью очереди.

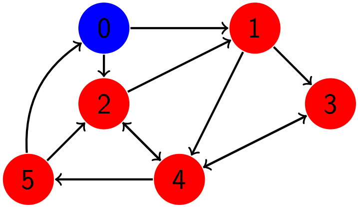
Алгоритм обхода.

- Поместить в очередь стартовую вершину, пометить её как просмотренную.
- Пока очередь непуста:
  - Извлечь из очереди первую вершину.
  - Добавить в очередь всех её непросмотренных соседей, помечая их как просмотренных.
- Если имеется непросмотренная вершина, повторить алгоритм, взяв её в качестве стартовой.

# Обход в ширину: псевдокод

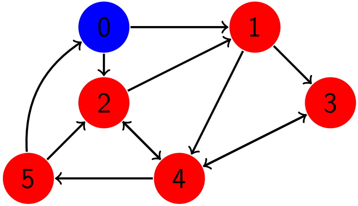
```
function BFS(G = ⟨V, E⟩, v)
    q = Queue()
    used = [False] * |V|                    ▷ индикатор посещения вершин
    distances = [∞] * |V|                   ▷ текущие расстояния до вершин
    answer = []                             ▷ вершины в порядке обхода
    q.push(v)
    used[v] = True
    d[v] = 0
    while q непуста do
        u = q.pop()
        answer.append(u)
        for w ∈ neighbours(u) do            ▷ перебираем соседей
            if not used[w] then
                q.push(w)
                used[w] = True
                d[w] = d[u] + 1
            end if
        end for
    end while
end function
return answer
```

# Обход в ширину: демонстрация

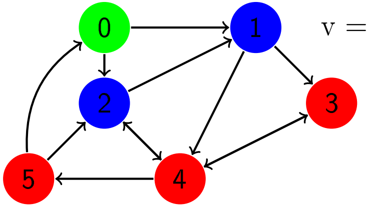



$q = [0]$

# Обход в ширину: демонстрация

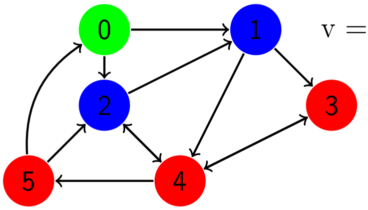



$$q = [0]$$

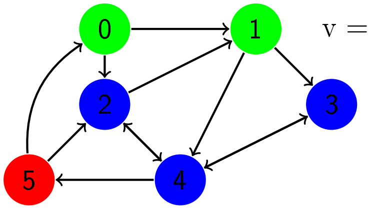



$$v = 0, d = [0, 1, 1, \infty, \infty, \infty], q = [1, 2]$$

# Обход в ширину: демонстрация

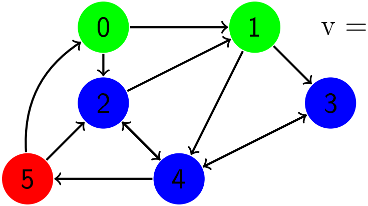



$v = 0, d = [0, 1, 1, \infty, \infty, \infty], q = [1, 2]$

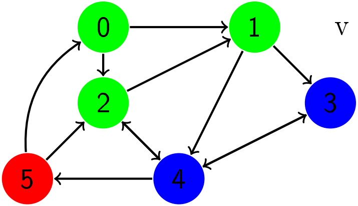



$v = 1, d = [0, 1, 1, 2, 2, \infty], q = [2, 3, 4]$

# Обход в ширину: демонстрация

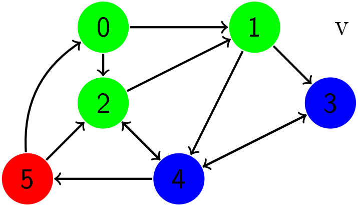



$$v = 1, d = [0, 1, 1, 2, 2, \infty], q = [2, 3, 4]$$

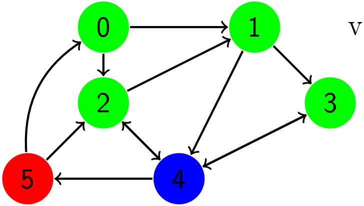



$$v = 2, d = [0, 1, 1, 2, 2, \infty], q = [3, 4]$$

# Обход в ширину: демонстрация

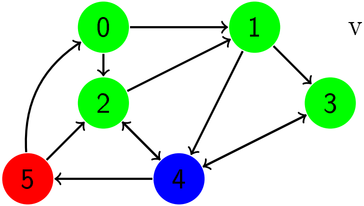



$$v = 2, d = [0, 1, 1, 2, 2, \infty], q = [3, 4]$$

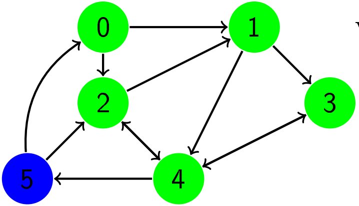



$$v = 3, d = [0, 1, 1, 2, 2, \infty], q = [4]$$

# Обход в ширину: демонстрация

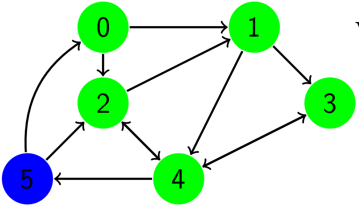



$$v = 3, d = [0, 1, 1, 2, 2, \infty], q = [4]$$

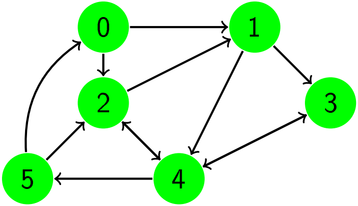



$$v = 4, d = [0, 1, 1, 2, 2, 3], q = [5]$$

# Обход в ширину: демонстрация




$$v = 4, d = [0, 1, 1, 2, 2, 3], q = [5]$$




$$v = 5, d = [0, 1, 1, 2, 2, 3], q = []$$

# Обход в ширину: анализ

- Алгоритм гарантированно завершается (каждая вершина посещается не более одного раза).

# Обход в ширину: анализ

- Алгоритм гарантированно завершается (каждая вершина посещается не более одного раза).
- Сложность алгоритма: $O(|E|)$ (каждое ребро рассматривается не более одного раза).

# Обход в ширину: анализ

- Алгоритм гарантированно завершается (каждая вершина посещается не более одного раза).
- Сложность алгоритма: $O(|E|)$ (каждое ребро рассматривается не более одного раза).
- Вершины помещаются в очередь в порядке возрастания их расстояния от исходной вершины $v$.

# Обход в ширину: анализ

- Алгоритм гарантированно завершается (каждая вершина посещается не более одного раза).
- Сложность алгоритма: $O(|E|)$ (каждое ребро рассматривается не более одного раза).
- Вершины помещаются в очередь в порядке возрастания их расстояния от исходной вершины $v$.
- Для любого $k$ можно найти момент, когда очередь состоит из вершин на расстоянии $k$, они помещаются в стек со значением $d[w] = k$.
- Алгоритм правильно вычисляет расстояние $d[w]$ до вершины $w$ от стартовой.

# Постановка задачи

- Часто нужно уметь вычислять расстояние от некоторой вершины графа до всех остальных:
  - Поиск кратчайшего географического пути.

# Постановка задачи

- Часто нужно уметь вычислять расстояние от некоторой вершины графа до всех остальных:
    - Поиск кратчайшего географического пути.
    - Поиск гипотезы наименьшего веса (наибольшей вероятности) в задачах вычислительной лингвистики.
    - Поиск слова наименьшего веса, принимаемого конечным автоматом и т.д.

# Постановка задачи

- Часто нужно уметь вычислять расстояние от некоторой вершины графа до всех остальных:
  - Поиск кратчайшего географического пути.
  - Поиск гипотезы наименьшего веса (наибольшей вероятности) в задачах вычислительной лингвистики.
  - Поиск слова наименьшего веса, принимаемого конечным автоматом и т.д.
- В случае рёбер стоимости 1 это делает алгоритм поиска в ширину, используя очередь.
- Существуют модификация, позволяющая обрабатывать рёбра длины 0 и 1 (двусторонняя очередь).

# Постановка задачи

- Часто нужно уметь вычислять расстояние от некоторой вершины графа до всех остальных:
  - Поиск кратчайшего географического пути.
  - Поиск гипотезы наименьшего веса (наибольшей вероятности) в задачах вычислительной лингвистики.
  - Поиск слова наименьшего веса, принимаемого конечным автоматом и т.д.
- В случае рёбер стоимости 1 это делает алгоритм поиска в ширину, используя очередь.
- Существуют модификация, позволяющая обрабатывать рёбра длины 0 и 1 (двусторонняя очередь).
- В случае рёбер произвольной длины логично использовать очередь с приоритетом.
- Это делает алгоритм Дейкстры.

# Алгоритм Дейкстры

- Пусть $d[u]$ — расстояние от исходной вершины $v$ до вершины $u$. В начале работы $d[u] = 0, d[v] = \infty$ для $v \neq u$.

# Алгоритм Дейкстры

- Пусть $d[u]$ — расстояние от исходной вершины $v$ до вершины $u$. В начале работы $d[u] = 0, d[v] = \infty$ для $v \neq u$.
- Будем поддерживать множество необработанных вершин $Q$, вначале оно содержит все вершины.

# Алгоритм Дейкстры

- Пусть $d[u]$ — расстояние от исходной вершины $v$ до вершины $u$. В начале работы $d[u] = 0, d[v] = \infty$ для $v \neq u$.
- Будем поддерживать множество необработанных вершин $Q$, вначале оно содержит все вершины.
- Пока множество необработанных вершин непусто:
  - Найти в нём вершину $u$ с минимальным расстоянием $d[u]$, удалить её из множества.

# Алгоритм Дейкстры

- Пусть $d[u]$ — расстояние от исходной вершины $v$ до вершины $u$. В начале работы $d[u] = 0, d[v] = \infty$ для $v \neq u$.
- Будем поддерживать множество необработанных вершин $Q$, вначале оно содержит все вершины.
- Пока множество необработанных вершин непусто:
  - Найти в нём вершину $u$ с минимальным расстоянием $d[u]$, удалить её из множества.
  - Для всех вершин $v \in \mathrm{neighbours}(u)$ обновить значение $d[v]$:

$$d[v] = \min\left(d[u] + c(u, v), d[v]\right)$$

# Алгоритм Дейкстры

- Пусть $d[u]$ — расстояние от исходной вершины $v$ до вершины $u$. В начале работы $d[u] = 0, d[v] = \infty$ для $v \neq u$.
- Будем поддерживать множество необработанных вершин $Q$, вначале оно содержит все вершины.
- Пока множество необработанных вершин непусто:
  - Найти в нём вершину $u$ с минимальным расстоянием $d[u]$, удалить её из множества.
  - Для всех вершин $v \in \mathrm{neighbours}(u)$ обновить значение $d[v]$:

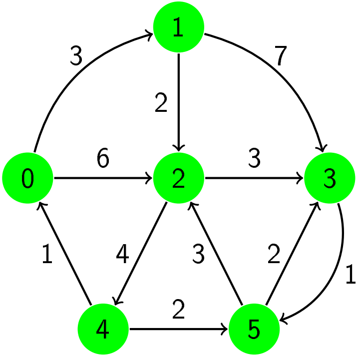
$$d[v] = \min\left(d[u] + c(u, v), d[v]\right)$$

**Утверждение**

После удаления вершины $v$ из множества обработанных вершин значение $d[v]$ не изменится.

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.

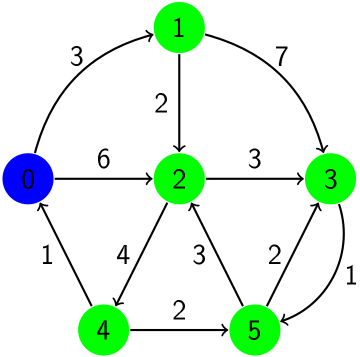



$(0, 0)$

$\mathbf{d} = [0, \infty, \infty, \infty, \infty, \infty]$

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.



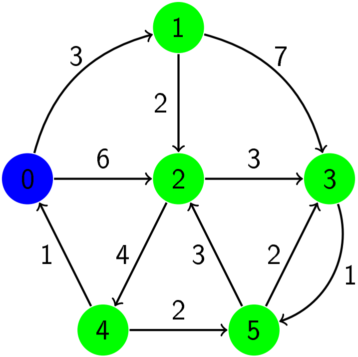

$(0, 0)$

$$d[1] = d[0] + d_{01} = 3$$
$$d[2] = d[0] + d_{02} = 6$$

$$\mathbf{d} = [0, 3, 6, \infty, \infty, \infty]$$

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.

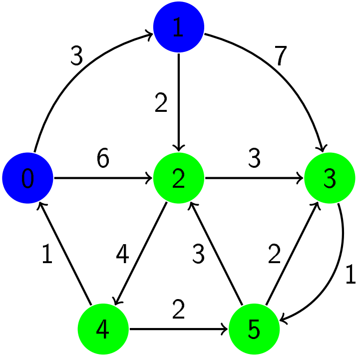



$(1,3),(2,6)$

$$\mathbf{d} = [0, 3, 6, \infty, \infty, \infty]$$

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.



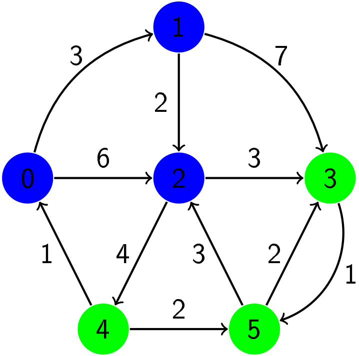

$(1, 3), (2, 5), (3, 10)$

$$d[2] = d[1] + d_{12} = 5 < 6$$
$$d[3] = d[1] + d_{13} = 10$$

$$\mathbf{d} = [0, 3, 5, 10, \infty, \infty]$$

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.



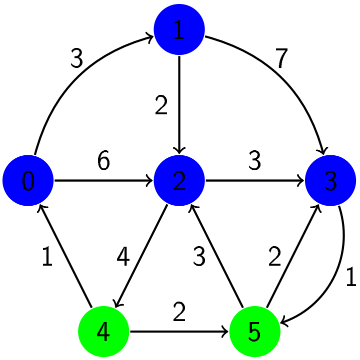

$(2, 5), (3, 8), (4, 9)$

$$d[3] = d[2] + d_{23} = 8 < 10$$
$$d[4] = d[2] + d_{24} = 9$$

$$\mathbf{d} = [0, 3, 5, 8, 9, \infty]$$

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.



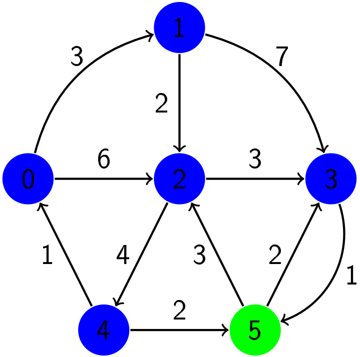

$(3, 8), (4, 9), (5, 9)$

$$d[5] = d[3] + d_{35} = 9$$

$$\mathbf{d} = [0, 3, 5, 8, 9, 9]$$

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.

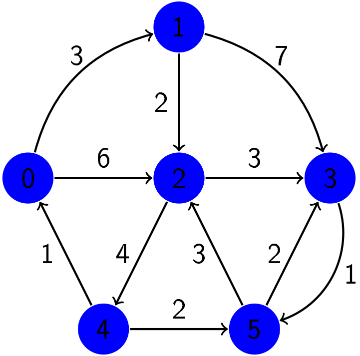



$(4, 9), (5, 9)$

$$\mathbf{d} = [0, 3, 5, 8, 9, 9]$$

# Алгоритм Дейкстры: демонстрация

Будем хранить в очереди с приоритетами номера вершин вместе с текущими расстояниями до них от вершины 0.




$(5, 9)$

$\mathbf{d} = [0, 3, 5, 8, 9, 9]$

# Алгоритм Дейкстры: анализ

- Для алгоритма Дейкстры удобно использовать очередь с приоритетом на основе невозрастающей пирамиды:
  - Извлечение минимума: $\mathrm{EXTRACT\text{-}MIN}$, сложность $O(\log|V|)$.

# Алгоритм Дейкстры: анализ

- Для алгоритма Дейкстры удобно использовать очередь с приоритетом на основе невозрастающей пирамиды:
  - Извлечение минимума: $\mathrm{EXTRACT\text{-}MIN}$, сложность $O(\log|V|)$.
  - Обновление значения $d(v)$: $\mathrm{DECREASE\text{-}KEY}$, сложность $O(\log|V|)$.
- Сложность алгоритма:
  - Минимум извлекается один раз для каждой вершины, суммарная сложность $O(|V|\log|V|)$.
  - Расстояние обновляется не более одного раза на каждом ребре, суммарная сложность $O(|E|\log|V|)$.

# Алгоритм Дейкстры: анализ

- Для алгоритма Дейкстры удобно использовать очередь с приоритетом на основе невозрастающей пирамиды:
  - Извлечение минимума: EXTRACT-MIN, сложность $O(\log|V|)$.
  - Обновление значения $d(v)$: DECREASE-KEY, сложность $O(\log|V|)$.
- Сложность алгоритма:
  - Минимум извлекается один раз для каждой вершины, суммарная сложность $O(|V|\log|V|)$.
  - Расстояние обновляется не более одного раза на каждом ребре, суммарная сложность $O(|E|\log|V|)$.
- Общая сложность $O((|V|+|E|)\log|V|)$ при условии, что мы можем по номеру вершины определять её позицию в пирамиде.