

АДРИАНОВ Н.М.  
ИВАНОВ А.Б.

# АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

ГРАФЫ. КРАТЧАЙШИЕ ПУТИ.  
МАКСИМАЛЬНОЕ ПАРОСОЧЕТАНИЕ.  
ПОТОКИ В СЕТЯХ

# КРАТЧАЙШИЕ ПУТИ ИЗ ОДНОЙ ВЕРШИНЫ

**Дано:**

- ориентированный граф  $G = (V, E)$ ;
- у ребер заданы длины  $l : E \rightarrow \mathbb{R}$ ;
- стартовая вершина  $s$ .

**Требуется:** найти расстояния от  $s$  до  $v$  для всех вершин  $v \in V$ .

Алгоритм Дейкстры решает эту задачу, но только если длины ребер неотрицательны.

Алгоритм Дейкстры использует очередь с приоритетами. Его сложность зависит от структуры данных, с помощью которой реализуют очередь. При использовании бинарной кучи – сложность  $O(E \log V)$ .

## ЦИКЛЫ ОТРИЦАТЕЛЬНОЙ ДЛИНЫ

Если в графе существуют ребра отрицательной длины, то могут быть циклы отрицательной длины.

Если из вершины  $s$  можно добраться до цикла отрицательной длины, то задача поиска кратчайших путей из  $s$  не имеет решения.

Нужен алгоритм, который будет правильно определять кратчайшие пути, если в графе нет отрицательных циклов, а они если есть - то алгоритм должен определять это.

# ОРИЕНТИРОВАННЫЙ АЦИКЛИЧЕСКИЙ ГРАФ (DAG)

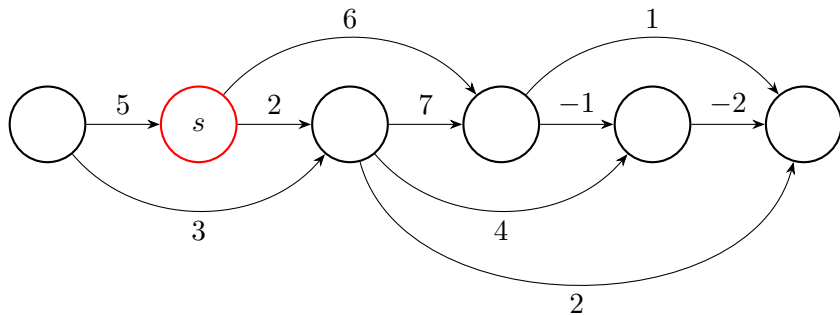
---

```
procedure DAGSSP( G, l, s )
  dist = массив размера V
  V = топологически отсортированные вершины G
  for v in V:
    dist[v] =  $+\infty$ 
  dist[s] = 0

  for u in V:
    for v in G.adj(u):
      if dist[v] > dist[u] + l[e]:
        dist[v] = dist[u] + l[e]
end
```

---

# ОРИЕНТИРОВАННЫЙ АЦИКЛИЧЕСКИЙ ГРАФ (DAG)



# АЛГОРИТМ БЕЛЛМАНА-ФОРДА

---

```
procedure BellmanFord( G, l, s )
  dist = массив размера V
  for v in V:
    dist[v] =  $+\infty$ 
  dist[s] = 0

  for i = 1,2,...|V|-1:
    for u in V:
      for v in G.adj(u):
        if dist[v] > dist[u] + l[e]:
          dist[v] = dist[u] + l[e]
end
```

---

# АЛГОРИТМ БЕЛЛМАНА-ФОРДА

---

```
procedure BellmanFord( G, l, s )
  dist = массив размера V
  for v in V:
    dist[v] =  $+\infty$ 
  dist[s] = 0

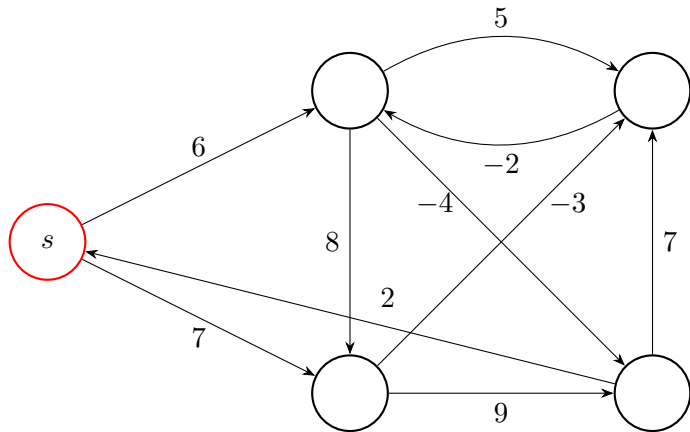
  for i = 1, 2, ..., |V|-1:
    for u in V:
      for v in G.adj(u):
        if dist[v] > dist[u] + l[e]:
          dist[v] = dist[u] + l[e]
end
```

---

Сложность  $O(V \cdot E)$

- Если отрицательных циклов нет, то в кратчайшем пути вершины не могут повторяться. Значит, длина кратчайшего пути  $\leq V - 1$ .
- Если сделать  $|V|$  итераций и на последней итерации меняется хотя бы одно расстояние – значит есть отрицательные циклы.
- Если на какой-то итерации изменений не происходит, то цикл можно прерывать.

# АЛГОРИТМ БЕЛЛМАНА-ФОРДА



## КОРРЕКТНОСТЬ АЛГОРИТМА БЕЛЛМАНА-ФОРДА

Идея: пусть в графе нет отрицательных циклов. Тогда для любой вершины  $v$  кратчайший путь  $p = (s, v_1, v_2, \dots, v_k)$ ,  $k < V$ . Любое начало этого пути - кратчайший путь к соответствующей вершине.

# ЗАДАЧА О САМОМ ДЛИННОМ ПУТИ

**Дано:**

- ориентированный граф  $G = (V, E)$ ;
- у ребер заданы длины  $l : E \rightarrow \mathbb{R}$ ;
- стартовая вершина  $s$ .

**Требуется:** найти самый длинный путь от  $s$  до  $v$  для всех вершин  $v \in V$ .

# КРАТЧАЙШИЙ ПРОСТОЙ ПУТЬ

**Дано:**

- ориентированный граф  $G = (V, E)$ ;
- у ребер заданы длины  $l : E \rightarrow \mathbb{R}$ ;
- стартовая вершина  $s$ .

**Требуется:** найти кратчайший простой путь от  $s$  до  $v$  для всех вершин  $v \in V$ .

# ВСЕ КРАТЧАЙШИЕ ПУТИ

**Дано:**

- ориентированный граф  $G = (V, E)$ ;
- у ребер заданы длины  $l : E \rightarrow \mathbb{R}$ .

**Требуется:** найти расстояния от  $u$  до  $v$  для всех вершин  $u, v \in V$ .

## ВСЕ КРАТЧАЙШИЕ ПУТИ

Алгоритм Дейкстры из каждой вершины:  $O(VE \log V)$

- $O(V^2 \log V)$ , если граф разреженный:  $E = O(V)$ ;
- $O(V^3 \log V)$ , если граф плотный:  $E = O(V^2)$ .

Алгоритм Форда-Беллмана из каждой вершины:  $O(V^2 E)$

- $O(V^3)$ , если граф разреженный:  $E = O(V)$ ;
- $O(V^4)$ , если граф плотный:  $E = O(V^2)$ .

Алгоритм Флойда – дальше.

# АЛГОРИТМ ФЛОЙДА

Пронумеруем вершины  $V = \{1, 2, \dots, n\}$ .

Динамика: ищем пути из  $i$  в  $j$  так, что в по пути проходим только через вершины с номерами не больше  $k$ .

---

```
procedure Floyd( G, l )
  A = 3-мерный массив (реально нужен только 2-мерный)
  for i = 1, ... n
    for j = 1, ... n
      A[i, j, 0] =
        0,          если  $i = j$ 
        l[i,j],     если  $(i,j) \in E$ 
         $+\infty$     иначе

  for k = 1, ... n
    for i = 1, ... n
      for j = 1, ... n
        A[i, j, k] = min(A[i, j, k - 1], A[i, k, k - 1] + A[k, j, k - 1])
end
```

---

## АЛГОРИТМ ФЛОЙДА

Q: Как определить наличие отрицательного цикла?

A: В конце работы алгоритма  $A[i, i, n] < 0$   
для некоторого  $i$ .

Q: Как восстановить кратчайший путь?

A: Дополнительно для каждой пары  $(i, j)$  надо хранить максимальный индекс вершины, входящей в кратчайший путь от  $i$  до  $j$ .

## ИЗБАВЛЕНИЕ ОТ ОТРИЦАТЕЛЬНЫХ ДЛИН РЕБЕР

Цель: избавиться от отрицательных ребер, чтобы можно было воспользоваться алгоритмом Дейкстры для решения задачи нахождения всех кратчайших путей.

1. Добавим новую вершину  $s$  и соединим ее со всеми вершинами графа  $G$  ребрами веса 0.
2. Запустим алгоритм Беллмана-Форда для вершины  $s$  и определим  $p_u =$  кратчайшее расстояние от  $s$  до  $u$ .
3. Зададим новую длину ребер:  $l'(u, v) = l(u, v) + p_u - p_v$ .

$$p_v \leq p_u + l(u, v) \quad \Rightarrow \quad l'(u, v) \geq 0$$

Для любого пути из  $s$  в  $t$  новая длина пути отличается на  $p_s - p_t$ .