


# Лекции по Алгоритмам и структурам данных. Часть 2

Конспекты   
Весна 2026

# Оглавление

<b>Лекция 1</b> .....	<b>2</b>
1.1. Графы с отрицательными ребрами .....	2
1.1.1. Форд-Белман с очередью .....	2
1.1.2. Через динамику .....	2
1.1.3. Отрицательные циклы .....	2
1.1.4. Флойд .....	3
1.1.5. Потенциалы .....	3
1.1.5.1. Джонсон .....	3
1.1.6. Транзитивное замыкание .....	3
<b>Лекция 3</b> .....	<b>4</b>
3.1. BST .....	4
3.1.1. За $\mathcal{O}(1)$ .....	4
3.1.2. AVL .....	4
3.2. Хеш-Таблицы .....	5
3.2.1. Открытая адресация .....	5

# Лекция 1

## 1.1. Графы с отрицательными ребрами

### 1.1.1. Форд-Белман с очередью

Пока можем улучшить ребро, улучшаем

```
while Run
  Run ← 0
  for e : a → b
    if da + wab < db
      db ← aa + wab
      Run ← 1
```

1. очередь с вершинами для которых поменялось расстояние

Примечание: Если внешний цикл сделал  $> n - 1$  итераций, то есть цикл отрицательного веса.

**Теорема 1.1.1.1:** Эта штука работает за  $O(n \cdot m)$ .

*Доказательство:* Пусть depth — глубина дерева минимальных путей. Тогда время работы  $\text{Time} = E \cdot \text{depth}$ . Где depth рандомный в среднем  $\log n$ . ■

### 1.1.2. Через динамику

$\text{dp}[k, v]$  — минимальный вес пути. Пересчитывем переход  $k \mapsto k + 1$  за  $O(E)$ .

Примечание:

$$\forall e : v \rightarrow u \quad \text{dp}[k + 1, v] = \min(\text{dp}[k + 1, v], \text{dp}[k, v] + w_{vu})$$

### 1.1.3. Отрицательные циклы

Примечание: Если делаем Форда-Белмана на очереди, то если сделали  $n$  итераций то существует отрицательный цикл. Однако нельзя сказать что если улучшили на последней итерации расстояние до  $v$ , то не значит что она в цикле, она достижима из него. Поэтому чтобы найти сам цикл надо пройти по дереву обхода обратно. Если найдем какой-то цикл то это и есть он.

*Доказательство:* Предположим что не закиклились. Тогда есть путь от  $s$  до  $v$ . Рассмотрим какое-то  $p[a]$  где  $e : a \rightarrow \cdot$ .

$$d[p[a]] + w_e \leq d[a] \implies \underbrace{d[s]}_0 + \underbrace{\omega(\text{path})}_{\leq n-1} \leq \underbrace{d[v]}_{n\text{-тая фаза}}$$

Покажем что этот цикл отрицательный. **Доделать**

$$\sum d + \omega(\text{cycle}) \leq \sum d$$

## 1.1.4. Флойд

$$d[i, j] = \begin{cases} w_{ij} \\ +\infty \end{cases}$$

**for**  $k$

```

| for  $i$ 
| | for  $j$ 
| | |  $d[i, j] \stackrel{\text{Relax}}{\leftarrow} d[i, k] + d[k, j]$ 

```

*Доказательство:*

- $k = 0$ . Очев
- $k \mapsto k + 1$ . Уже знаем минимум для всех  $k$ . Очев? **Но это не точно**

■

*Примечание:*

- Отрицательные циклы если есть  $d[u, v] < 0$
- Восстановление пути. Храним следующую вершину в пути  $p[i, j]$ . Изначально  $p[i, j] = j$ . Если обновляем через  $k$ , то  $p[i, j] = p[i, k]$ . Потом идем по  $i = p[i, j]$ .

## 1.1.5. Потенциалы

Можем поменять веса ребер так для какого-то  $e : a \rightarrow b$ .  $w_e \mapsto w_e + p[a] - p[b]$ . Посмотрим как изменятся веса путей  $w(\text{path}) \mapsto w(\text{path}) + p[s] - p[t]$ . Где  $p$  — **потенциалы** — любые вещественные числа. Можно заметить что минимальный путь останется минимальным.

Как подобрать  $p$  так чтобы веса стали неотрицательными. Найдем расстояние из фиктивной вершины до всех остальных вершин Форд-Белманом. Возьмем расстояния  $d$  как потенциалы.

*Доказательство:* Покажем что  $w_e + d[a] + d[b] \geq 0$ . Если оказалось что  $w_e + d[a] < d[b]$  то это значит что  $d[b]$  посчитано неверно. Противоречие. ■

## 1.1.5.1. Джонсон

Чтобы найти расстояния от каждой вершины до каждой то Флойдом  $V^3$ . Но можем Ф.Б найти потенциалы, избавиться от отрицательных ребер и запустить Дейкстру  $n$  раз:  $F.B. + V \cdot \text{Dijkstra} = VE + V^2 \log V \leq V^3$

## 1.1.6. Транзитивное замыкание

Можно решать Флойдом:

```

for  $k$ 
| for  $i$ 
| | for  $j$ 
| | | if  $c_{ik} \wedge c_{kj}$ 
| | | |  $c_{ij} \leftarrow T$ 

```

Пооптимизируем. Заметим что делаем  $c[j] = c[k]$ . Можно сделать на битсетах.

# Лекция 3

## 3.1. BST

**Доделать**

*Примечание:*  $\text{next}(v)$  — спускаемся максимально влево

### 3.1.1. За $\mathcal{O}(1)$

Можем делать Find за  $\mathcal{O}(1)$  храня карту из значений в вершину. Для минимума/максимуму храним соответственно ссылки на них.

Как делать Next за  $\mathcal{O}(1)$ : храним для каждой вершины Next и Prev. Обновляем при изменении, получается двусвязный список.

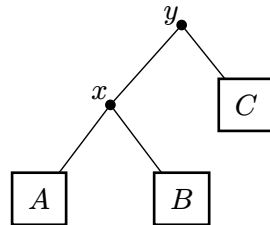
*Примечание:* Нельзя сделать Add за  $\mathcal{O}(1)$ , иначе смогли бы отсортировать массив за  $\mathcal{O}(n)$ .

### 3.1.2. AVL

**Определение 3.1.2.1:** Дерево обладает свойством AVL если для любой вершины, глубина левого поддерева отличается от глубины правого не более чем на 1.

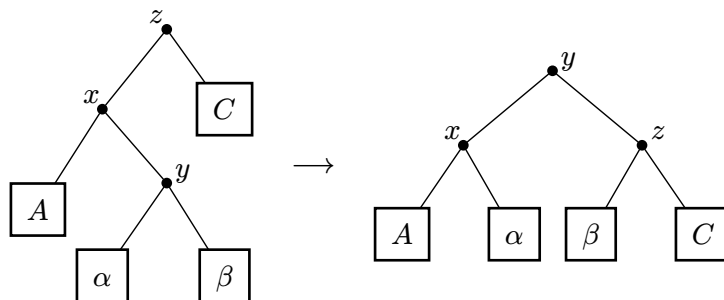
При добавлении поднимаемся на верх и делаем повороты.

*Примечание:*



Если после добавления в поддерево  $B$  получились деревья  $h_A = h, h_B = h + 1, h_C = h$ , то  $h_x = h + 2$  и нужно сделать поворот. Однако после поворота проблема останется.

Выделить корень  $z$  в поддереве  $B$ , и вынести его наверх,  $x$  слева,  $y$  справа — **большое вращение**.



**Теорема 3.1.2.1:** Всегда делаем не больше одного вращения

*Примечание:* При удалении можем пометить вершину как удаленную и что-то делать потом с ней **Доделать**

## 3.2. Хеш-Таблицы

Заводим массив массивов и мапим индексы в нем  $i \rightarrow i \bmod N$ , где  $N$  размер массива.

*Примечание:* Вместо фиксированного  $N$  берем рандом от  $N$  до  $2N$ . Тогда вероятность коллизии  $i \bmod N' \neq j \bmod N' \cdot i - j \mid N'$  с вероятностью(?)  $\log$ .

**Доделать**

### 3.2.1. Открытая адресация

*Примечание:* При удалении помечаем как “удаленный”