1 Ford-Bellman Algorithm

Обозначения:
n — количество вершин, m — количество рёбер.

1.1 Задачи

1. $w_e \geqslant 0$
   Dijkstra Algorithm
   $\mathcal{O}(n^2)$
   $\mathcal{O}(m \log n)$
   $\mathcal{O}(m \log_{\frac{m}{n}} n)$
   $\mathcal{O}(m + n \log n)$
   $\mathcal{O}(m + n \log C)$
   $\mathcal{O}(m + n\sqrt{\log C})$

   Если $w_e \in \mathbb{Z}$:
   Для неориентированного графа: $\mathcal{O}(n + m)$
   Для ориентированного графа: $\mathcal{O}(m + n \log\log \min(n, C))$
   $\exists$ алгоритм для ориентированного графа за $\mathcal{O}(n + m) \Rightarrow \exists$ IntegerSorting за $\mathcal{O}(n + m)$

2. $w_e \in [-N; +\infty] \cap \mathbb{Z}$:
   1958' Ford-Bellman $\mathcal{O}(nm)$
   1983' Gabow $\mathcal{O}(mn^{\frac{3}{4}} \log nN)$
   1989' Gabow and Tarjan $\mathcal{O}(mn^{\frac{1}{2}} \log nN)$
   1993' Goldberg $\mathcal{O}(mn^{\frac{1}{2}}(\log N + 1))$

1.2 Алгоритм

Можем найти следующую информацию:

$$\left[\begin{array}{l} d[v] \\ \nexists\, path \\ \nexists\, minimum\, path \Rightarrow \exists\, negative\, cycle\ (\mathcal{O}(n)) \end{array}\right.$$

start — стартовая вершина.

```
d[0,start] = 0
for v = 0..n-1:
        if (v != start)
                d[0,v] = +INF
for k = 0..n-1:
{
        for v = 0..n-1:
                d[k+1,v] = d[k,v]
```

```
        for e = 0..m-1: // e : a --w--> b
                if (d[k+1,b] > d[k,a] + w)
                {
                        d[k+1,b] = d[k,a] + w
                        if (k == n-1)
                                printf("negative cycle detected")
                }
}
```

$d[k, v]$ — минимальный путь из не более, чем k рёбер от $start$ до v.
На самом деле индекс k не нужен.

```
d[start] = 0
for v = 0..n-1:
        if (v != start)
                d[v] = +INF
for k = 0..n-1:
        for e = 0..m-1:  // e : a --w--> b
            if (d[b] > d[a] + w)
            {
                    d[b] = d[a] + w
                    if (k == n-1)
                            printf("negative cycle detected")
            }
```

$d[v]$ на k-ой итерации — учтены все пути длины $\leqslant k$ и ещё какие-то (из-за if-а пути могут получиться более длинными).
$d[v] = +\infty \Rightarrow \nexists\, path$

```
if (k == n-1)
        is[b] = 1
```

$is[v]$ — пометка о том, что до вершины $\nexists$ кратчайшего пути.
Достижимые из всех помеченных вершин не имеют кратчайшего пути.
На каждом отрицательном цикле хотя бы одна вершина помечена.
Два способа:

1. Если сделаем внешних итераций в 2 раза больше, то в конце отметим все нужные вершины.

   ```
   if (k == n-1 || is[a])
           is[b] = 1
   ```

2. После алгоритма Форда-Беллмана запустим dfs из всех помеченных вершин.

```
for v = 0..n-1:
        if (is[v] && !used[v])
                dfs(v)
```

$T = \mathcal{O}(nm)$
$M = \mathcal{O}(m)$

1.3 Оптимизации

1. $\mathcal{O}(km)$, где k — максимальное число рёбер в кратчайшем пути. В худшем случае $k = n$.

2. Random Shuffle рёбер
   $E = \frac{n}{2}$ — матожидание
   $E = 1 + \frac{1}{2}E$, $\frac{1}{2}$ — вероятность того, что прошли второе ребро после первого, E — матожидание от следующей вершины.
   $E = 1 + \frac{1}{2}\left(1 + \frac{1}{2}\left(1 + \frac{1}{2}\left(...\right)\right)\right) = 1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^k} + \cdots = 2$
   $E = 2 \Rightarrow$ нужно $\frac{n}{2}$ итераций.
   Асимптотика: $E(T) = \frac{km}{2}, T \leqslant km$, где k — длина наидлиннейшего кратчайшего пути.

3. Добавляем break, если на итерации не релаксировано ни одно ребро.

   ```
   run = 1
   while (run):
   {
           run = 0
           for e = 0..m-1   // e : a ---w---> b
               if (d[b] > d[a] + w)
               {
                       d[b] = d[a] + w
                       run = 1
               }
   }
   ```

   Случайный граф — веса ребер равновероятно выбираются на $[L..R]$, каждое ребро есть в графе с вероятностью p.

   Замечание. В случайном графе k мало.

4. Убираем оптимизацию 1 про random_shuffle, оставляем оптимизацию 3 про break.
   Рассматриваем только ребра из вершин, расстояние до которых поменялось на прошлой итерации.

   Идея:

   ```
   run = 1
   A={start}
   ```

```
while (run):
{
        run = 0, B = NULL
        for e = 0..m-1   // e : a ---w---> b
            if (d[b] > d[a] + w)
            {
                    d[b] = d[a] + w
                    run = 1
                    B += {b}
            }
        A = B
}
```

Реализация:

```
queue<int> q;
q.push(start)
in_queue[start] = 1
while (q.size()):
{
        int v = q.top(), q.pop()
        in_queue[v] = 0
        for e = 0..m-1   // e : a ---w---> b
            if (d[b] > d[a] + w)
            {
                    d[b] = d[a] + w
                    if (!in_queue[b])
// можем не добавлять b в очередь,
//потому что b уже лежит там ⇒ всё равно обработаем позже
                    {
                            q.push(b)
                            in_queue[b] = 1
                    }
                    run = 1
            }
}
```

5. Ацикличные графы

   Мы умеем искать кратчайшие пути для ацикличных графов за $\mathcal{O}(n+m)$ с помощью TopSort.
   Выделим компоненты сильной связности, будем обрабатывать их в порядке Topsort-а. Для отдельной компоненты связности, когда до нее дошла очередь, запустим Ford-Bellman, который будет работать за $\mathcal{O}(V'E')$, где V' — количество вершин в данной

компоненте сильной связности, E' — количество рёбер в данной компоненте сильной связности.

6. Levit

   Есть очередь, как и в оптимизации 3.

```
if (w <= 0 && is[b] == 0)
{
        is[b] = 1
        q.push_front(b)
}
```

   Асимптотика: $\mathcal{O}(nm)$

1.4 Поиск отрицательного цикла

```
for k = 0..n-1:
        for e = 0..m-1:   // e : a ---w---> b
            if (d[b] > d[a] + w)
            {
                    d[b] = d[a] + w
                    p[b] = e
                    if (k == n-1)
                            printf("negative cycle detected")
            }
```

Лемма 1

$\forall\, b\ d[b] \geqslant d[p[b].a] + p[b].w$

$b \xleftarrow{p[b].w} p[b].a$

Посмотрим на момент, когда мы сохранили $p[b]$ последний раз. В этот момент было равенство: $d[b] = d[p[b].a] + p[b].w$

После этого $d[b]$ не менялось, а $d[p[b].a]$ могло уменьшиться. Значит, неравенство действительно выполнено.

Лемма 2

Если мы пойдем по ссылкам, начиная с какой-то вершины, то зациклимся.

Пусть это не так.

Идём до вершины, у которой нет ссылки. Очевидно, это может быть только вершина $start$. Причём, если у неё нет ссылки, значит $d[start] = 0$.

$b \xleftarrow{w_1} a_1 \xleftarrow{w_2} a_2 \xleftarrow{w_3} \cdots \leftarrow start$

$d[b] \geqslant d[a_1] + w_1 \geqslant d[a_2] + w_1 + w_2 \geqslant \cdots \geqslant d[start] + \sum\limits_i w_i$

$d[b] \geqslant \sum\limits_i w_i$

$d[b] <$ все пути из $\leqslant n-1$ ребра, т.к. на n-ой итерации расстояние улучшилось.

$d[b] \geqslant$ конкретный путь из $\leqslant n-1$ ребра. Путь: $start, \ldots, a_2, a_1, b$.

Лемма 3

Любой цикл по ссылкам отрицателен.

Просумммируем неравенство леммы 1 по всем ребрам цикла.

$d[v] \geqslant d[p[v].a] + p[v].w$

$d[b] \geqslant d[p[b].a] + p[b].w \geqslant d[p[p[b].a].a] + p[p[b].a].w + p[b].w \geqslant \cdots \geqslant d[b] + \sum\limits_e w_e$

$0 \geqslant \sum\limits_e w_e$

Посмотрим на последнее прорелаксированное ребро в цикле. Пусть это ребро (v, u). Следующая по циклу после u — вершина w. Тогда расстояние $d[u]$ уменьшилось, значит по ребру (u, w) неравенство леммы 1 выполнено строго, а значит и цикл строго отрицательного веса.

2 Два указателя

x_n, w_n – координата и вес n-й точки $(x_i \leq x_{i+1}, w_i \geq 0)$

$[n, k]$ – состояние динамики, первые n точек разбили на k отрезков

$p_{n,k}$ – граница последнего отрезка

$q_{n,k}$ – центр последнего отрезка

$o_{l,r}$ – оптимальный центр отрезке $[l, r]$

Алгоритм выбора $o_{l,r}$ – самая левая точка, что $\sum[l, o_{l,r}] \geq \sum(o_{l,r}, r]$

Цель рассуждения

Доказать, что $p_{n,k-1} \leq p_{n,k} \leq p_{n+1,k}$

Последовательность утверждений:

Лемма 1

$o_{l,r-1} \leq o_{l,r} \leq o_{l+1,r}$

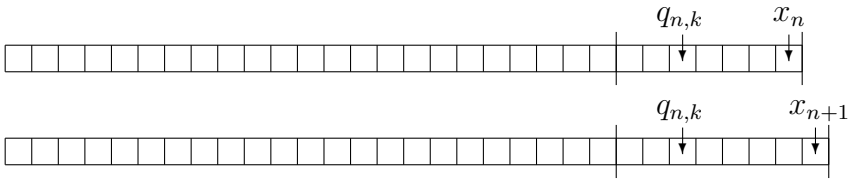
1) $o_{l,r-1} \leq o_{l,r}$ Доказательство: $\sum(o_{l,r}, r] \geq \sum(o_{l,r}, r-1]$, значит мы точно не будем двигать $o_{l,r-1}$ правее чем $o_{l,r}$

2) $o_{l,r} \leq o_{l+1,r}$ Доказательство: $\sum[l, o_{l,r}] \geq \sum[l+1, o_{l,r}]$, значит мы точно не будем двигать $o_{l+1,r}$ левее чем $o_{l,r}$

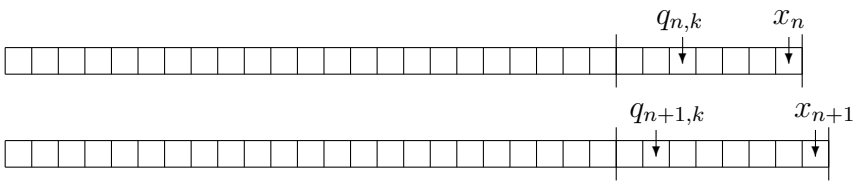
Лемма 2

$q_{n+1,k} \geq q_{n,k}$

Доказательство: перейдем от оптимального (n, k) разбиения к $(n+1, k)$ разбиению, не поменяв положение центра последнего отрезка $q_{n,k}$, функция увеличилась на $\mathbb{X} = w_{n+1}(x_{n+1} - x_{q_{n,k}})$



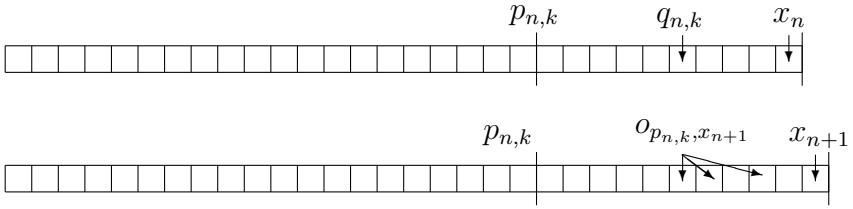
Теперь перейдем к оптимальному $(n+1,k)$ разбиению, функция уменьшилась.Теперь перейдем к (n,k) разбиению, функция уменьшилась на $\mathbb{Y} = w_{n+1}(x_{n+1} - x_{q_{n+1,k}})$



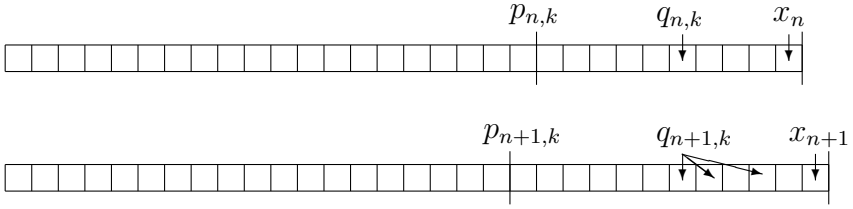
Если $q_{n+1,k} < q_{n,k}$, то $\mathbb{Y} > \mathbb{X}$, и, мы получаем противоречие с предположением, что исходное разбиение (n,k) было оптимальным.

Лемма 3

$p_{n+1,k} \geq p_{n,k}$. Доказательство: Возьмем оптимальное разбиение для (n,k), теперь возьмем разбиение для $(n+1,k)$, такое что $p_{n,k} = p_{n+1,k}$ при этом $q_{n,k} \leq q_{n+1,k}$ из Леммы 2.



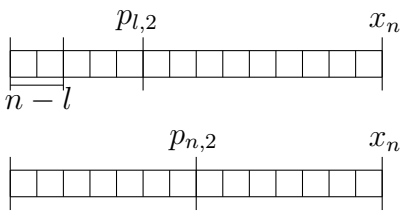
Пусть существует лучшее разбиение, такое что у него $p_{n+1,k} < p_{n,k}$



Тогда из Леммы 1 следует, что $o_{p_{n,k},xn+1} \geq q_{n+1,k}$, значит $w_{n+1}(x_{n+1} - x_{q_{n+1,k}}) \geq w_{n+1}(x_{n+1} - x_{o_{p_{n,k},x_{n+1}}})$, противоречие с тем, что изначально у нас было оптимальное разбиение для (n,k)

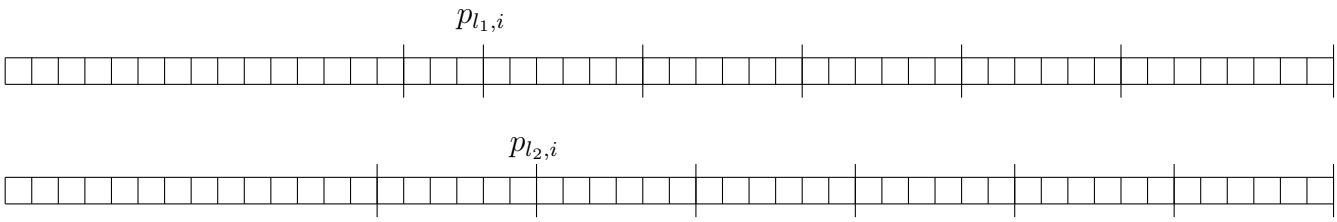
Лемма 4

$p_{n,k} \geq p_{n,k-1}$
Пусть $k = 3$ и $p_{n,k} < p_{n,k-1}$.
(ось x направлена справа налево)
l – длина двух правых отрезков из первого разбиения



Смотрим на правые два отрезка в обоих разбиениях, видим противоречие с Леммой 3, т.к. $l < n$, значит $p_{l,2} \leq p_{n,2}$.

Теперь пусть k – произвольное и $p_{n,k} < p_{n,k-1}$. Т.к. в первом разбиении больше отрезков, в какой-то момент получится такая ситуация, что i правых отрезков первого разбиения, меньше чем i правых отрезков второго разбиения. Рассмотрим первую такую ситуацию.
l_1 – длина i правых отрезков первого разбиения
l_2 – длина i правых отрезков второго разбиения.
(ось x направлена справа налево)



Получается противоречие: по Лемме 3 из того, что $l_1 < l_2$ следует, что $p_{l_2,i} \geq p_{l_1,i}$