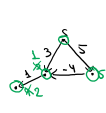
Кратчайшие пути во взвеш. графах



① Граф $w_e \geq 0$. Алгоритм Дейкстры (Dijkstra)
$s \to$ расст. до всех вершин

```
d=[+oo]*n; q = PriorityQueue()
d[s]=0; q.push((d[s],s))
```

```
addEdge(a, b, w):
    edges[a].append((b,w))
```

```
while q is not empty:
    d_cur, v = q.popMin()
    for u,w in edges[v]:
        if d[u] > d[v]+w:
            d[u] = d[v]+w
            q.changeKey(u, d[u])   # если вершина u не была в q, иначе создаём вр
```

Th. $\forall w$ будет обработана (т.е. достана из q) ≤ 2 раза; в момент обработки $d[v] = \rho(s,v)$. У необр. вершин $d[v] = \rho(s,v) = +\infty$.

Д-во

1) $\forall v \in V$ в любой момент $d[v] \geq \rho(s,v)$.
2) Обработ. верш. v.
От противного. Пусть $d[v] > \rho(s,v)$.
p – кр. путь $s \to v$.
β – первая вершина в p, кот. не обр.
α – предыдущая в p
$\rho(s,v) = \rho(s,\beta) + \rho(\beta,v)$

α – обработана. $d[\alpha] = \rho(s,\alpha)$. (по предп. индукции)

$\rho(s,\beta) \leq d[\beta] \leq d[\alpha] + w_{\alpha\beta} = \rho(s,\beta)$

$$d[v] > \rho(s,v) = \rho(s,\beta) + \underbrace{\rho(\beta,v)}_{\geq 0} \geq d[\beta]$$

Получается, что все $w_e \geq 0$

Итого, $d[v] > d[\beta]$. Но v была необр. вершиной с min $d[v]$! (?!)

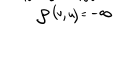
Инд. гипотеза: пусть i-я обр. вершина – v_i. Тогда когда её достают, $d[v_i] = \rho(s,v_i)$.

База: $i=1$. $v_1 = s$. $\rho(s,s) = 0$, $d[s] = 0$

3). Пусть обработали вершину v.
$d[v] = \rho(s,v)$ на последок. Улучшений $d[v]$ больше не будет. Значит, в очереди больше не встретится.

■

Rem. Если w_e бывают отриц., то в G нет отр. циклов. Тогда алг. Дейкстры корректен, но может обработать вершину много раз.



Реализации

① PriorityQueue – массив

```
d=[+oo]*n; used=[False]*n
d[s]=0
while True:                                  <- n итераций
    v=-1
    for i in range(n):
        if not used[i] and (v==-1 or d[i]<d[v]):   } <- O(n)
            v=i
    if v==-1: break
    used[v]=True
    for u,w in edges[v]:                     } суммарно за O(m)
        if d[u] > d[v]+w:
            d[u] = d[v]+w
```

$T = O(n^2 + m) = O(V^2 + E)$

② PriorQueue – куча

```
1  d=[+oo]*n; q = PriorityQueue()
2  d[s]=0; q.push((d[s],s))
3  while q is not empty:                     <- n итераций
4      d_cur, v = q.popMin()                 <- O(log n)
5      for u,w in edges[v]:                  <- m итераций суммарно
6          if d[u] > d[v]+w:
7              d[u] = d[v]+w
8              q.changeKey(u, d[u])   # если вершина u не была в q, иначе создаём вр   <- O(log n)
```

② .1 changeKey – честная ум. кучка в куче

```
7  d[u] = d[v]+w
8  siftUp(pos[u])
```

Time $= O(m \log n) = O(E \log V)$

② .2 не изменять, а добавл. новое

```
7  d[u] = d[v]+w
8  q.push((d[u], u))
```

Time $= O(m \log m) = O(E \log E)$

```
4:  d_cur, v = q.popMin();  if d_cur > d[v]: continue
```

② .3 не куча, а сбалансированное дерево поиска (set в C++)

Time $= O(m \log n) = O(E \log V)$

①: $O(V^2+E)$	Разреж. графы: $E \ll V^2$ – лучше ② способ
②.1: $O(E \log V)$	Плотные графы: $E \sim V^2$ – лучше ① способ