Построение и анализ алгоритмов

Лекция 11.2

Раздел: Алгоритмы на графах

Тема лекции:

Кратчайшие пути в графе

Кратчайшие пути в графе

Путь p в графе $G = (V, E)$:

$p: u \Rightarrow v \equiv u = u_1 \to u_2 \to u_3 \to \ldots \to u_k = v$

Взвешенный граф $G = (V, E, W)$,

где $w(i, j) = w(v_i, v_j)$ - произвольны.

Длина пути:

$$d_p[u,v] = \sum_{i=1}^{k-1} w(v_i, v_{i+1}) = \sum_{(v' \to v'') \in p} w(v', v'')$$

$p: u \Rightarrow v$, $p \in P(u, v)$ – множество путей между u и v

$$d(u,v) = \min_{p \in P(u,v)} d_p[u,v] - \text{"расстояние"}$$

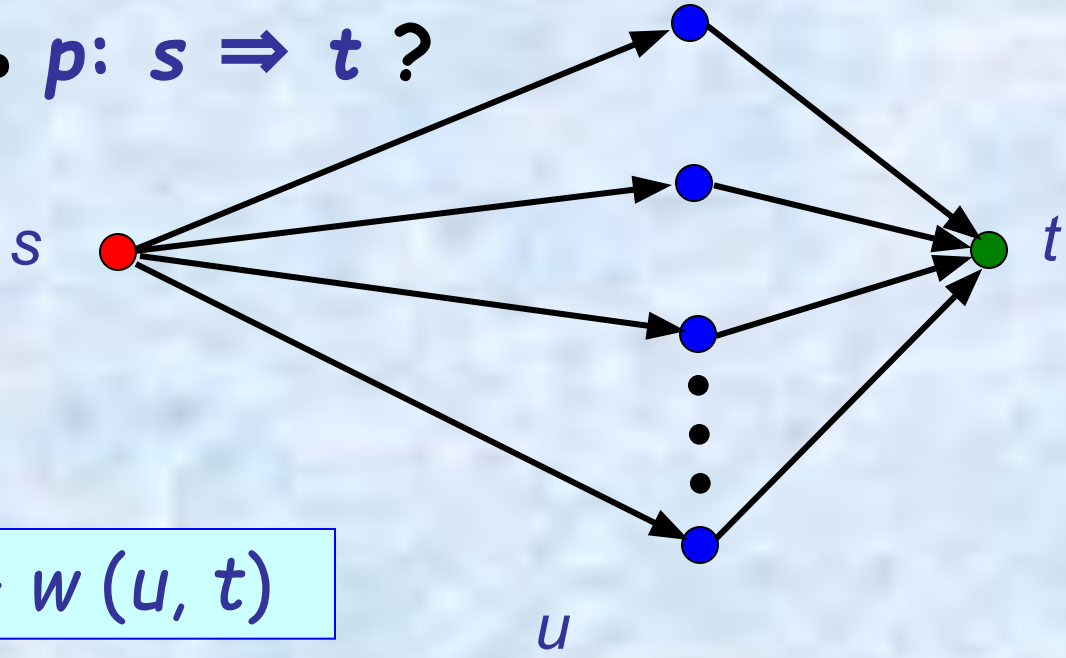
Приложения… $W(u, v) = -\log_2 \text{Prob}(u, v)$

Вычисление расстояний и нахождение путей

Пусть вычислены все расстояния $d(u, v)$,

в т.ч. $d(s, t)$.

Как найти путь p: $s \Rightarrow t$?

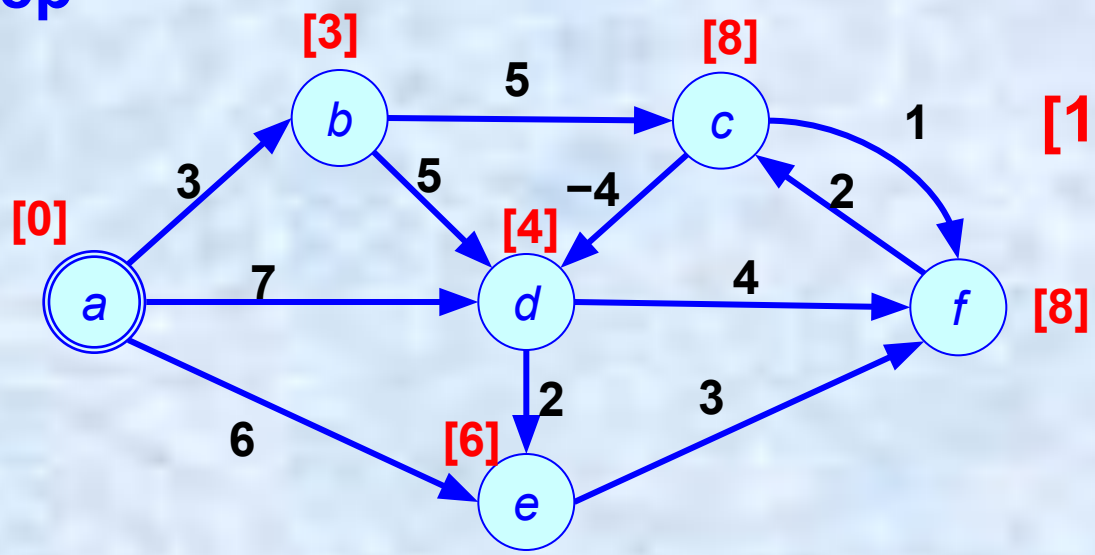


$$d(s, t) = d(s, u) + w(u, t)$$

Часть кратчайшего пути тоже кратчайший путь!

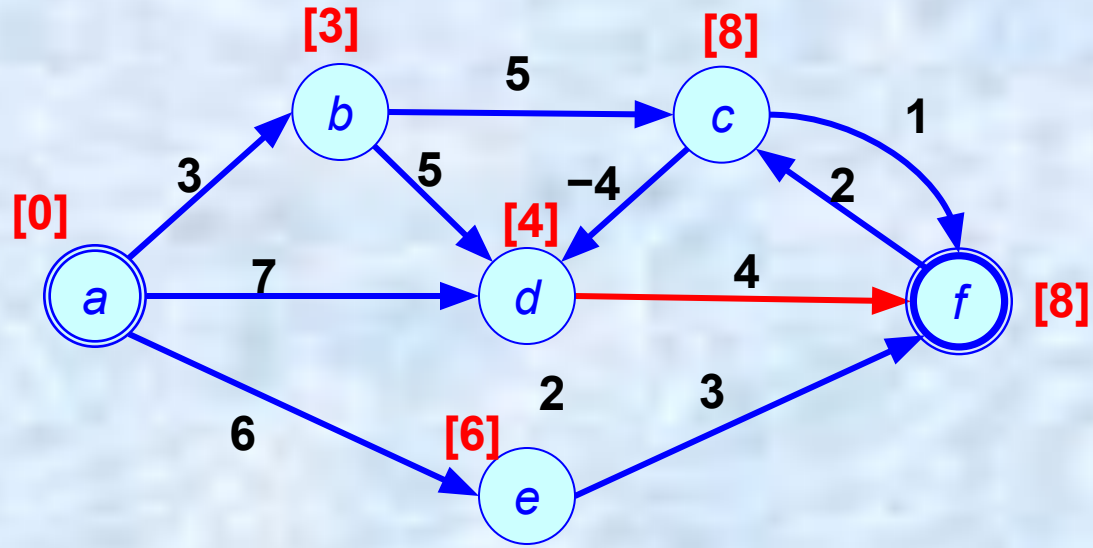
$v_1 \rightsquigarrow v_i \rightsquigarrow v_j \rightsquigarrow v_k \quad (1 \le i \le j \le k)$

Пример



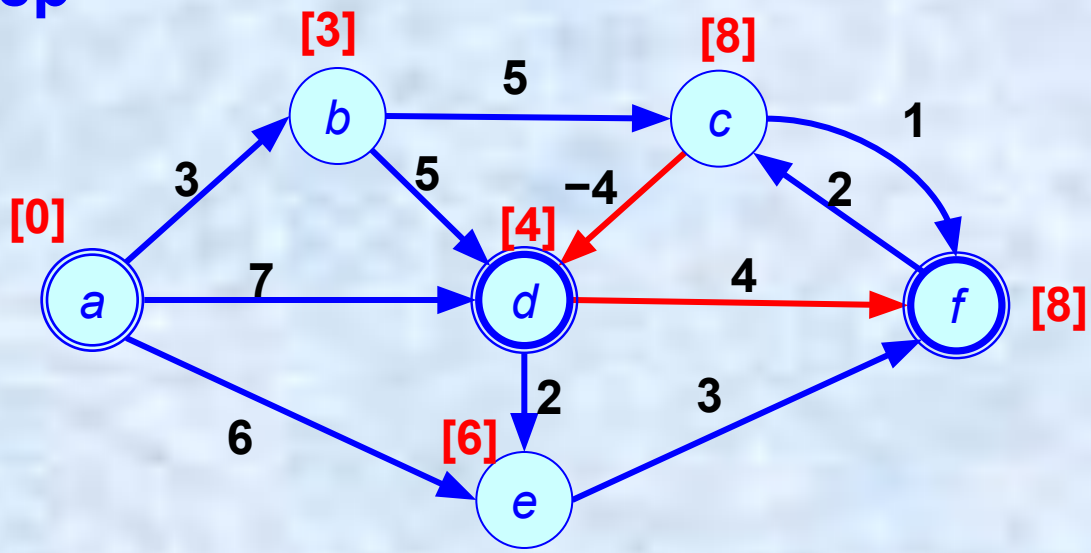
2 – вес ребра $w(i, j)$

[1] – расстояние $d(u,v)$

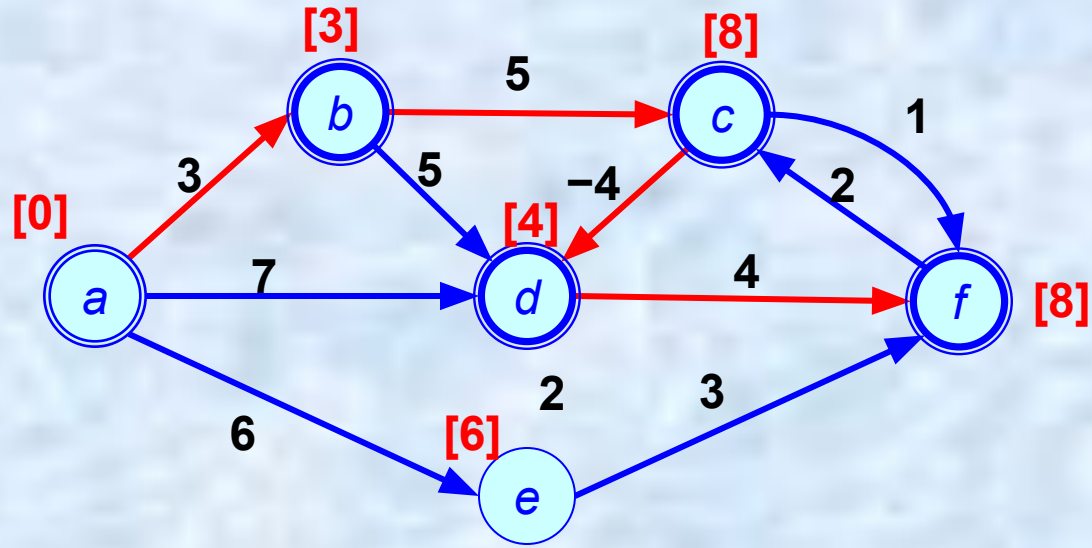


[8] = 4 + [4]

Пример



[4] = – 4 + [8]



[8] = 5 + [3]

[3] = 3 + [0]

Алгоритм нахождения кратчайшего пути по расстояниям между вершинами

```
/*Дано:
s, t – фиксированные вершины; s, t ∈V;
DL [ * ] – расстояния от вершины s до каждой вершины
   графа;          DL[v]=d(s,v);
W[u, v] – матрица весов ребер, u, v ∈V;
Результат: St – стек, содержащий кратчайший путь от s до
   t,
т.е. последовательность вершин s=v0, v1, … , vk=t
*/
Create(st);  st ← t; v = t;
while (v != s) {
   u = вершина, для которой DL[v]=DL[u]+W[u,v];
   St ← u;
   v=u;
 }
```

Алгоритм нахождения кратчайшего пути продолжение

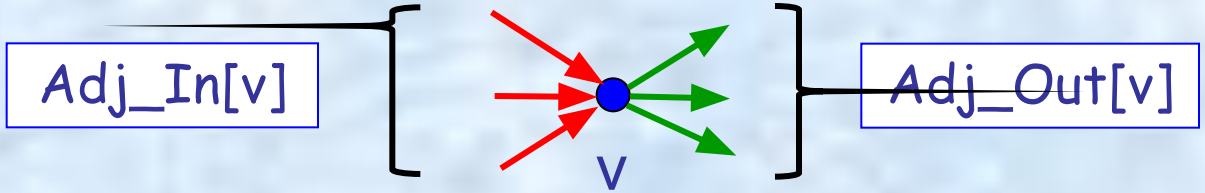
Детализация строки

u = вершина, для которой DL[v]=DL[u]+W[u,v];

```
Встать в начало Adj_In[v];
do
    u = очередной элемент из Adj_In[v];
while  (d (s, t) != d (s, u) + w [ u, t ])
```

Adj_In[v] – список смежности входящих ребер

Adj_Out[v] – список смежности исходящих ребер



Сложность

$O(m)$

Задачи вычисления длин кратчайших путей (расстояний)

1. *Стартовая* вершина s фиксирована, конечная вершина t фиксирована. Найти $d(s, t)$.
2. *Стартовая* вершина s фиксирована. Найти $d(s, v)$ для $\forall\ v \in V \setminus s$.
3. Найти $d(s, t) : \forall\ s, t \in V\ \&\ (s \neq t)$. Т.е. найти расстояния *между всеми* парами вершин.

Замечание: неизвестен ни один алгоритм решения задачи 1, который был бы более эффективным, чем известные алгоритмы, решающие задачу 2

Задачи вычисления расстояний от фиксированной вершины

Общая схема большинства алгоритмов:

рассматриваются временные пометки вершин $dl[v]$, которые являются *верхними ограничениями* для расстояний: $dl[v] \geq d(s, v)$.

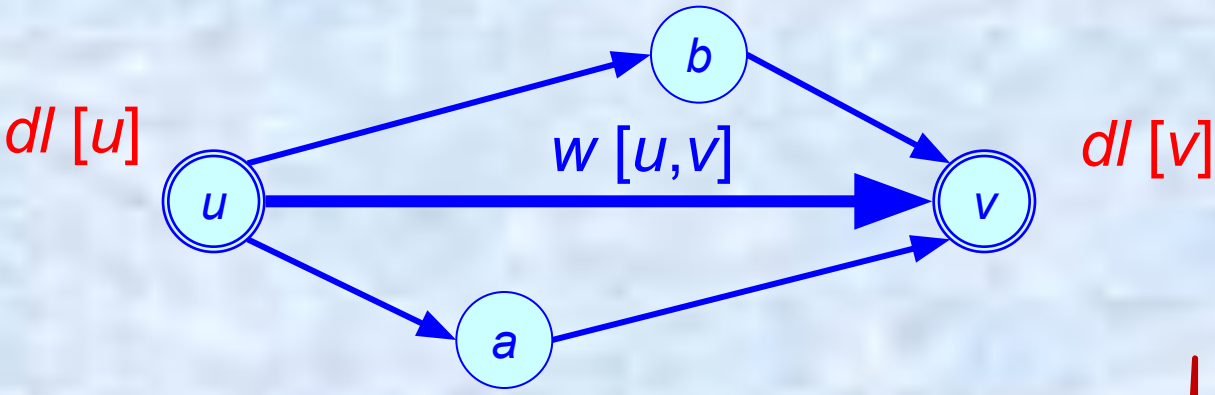
Улучшение оценки $dl[v]$:

```
void Relax (vert u, vert v)
// релаксация (ослабление) ребра u → v
{ if (dl [v] > dl [u] + W [ u, v ] )  dl [v] = dl [u] + W [ u, v ];
} //Relax
```

т.е. $dl[v] := \min(dl[v], dl[u] + W[u, v])$

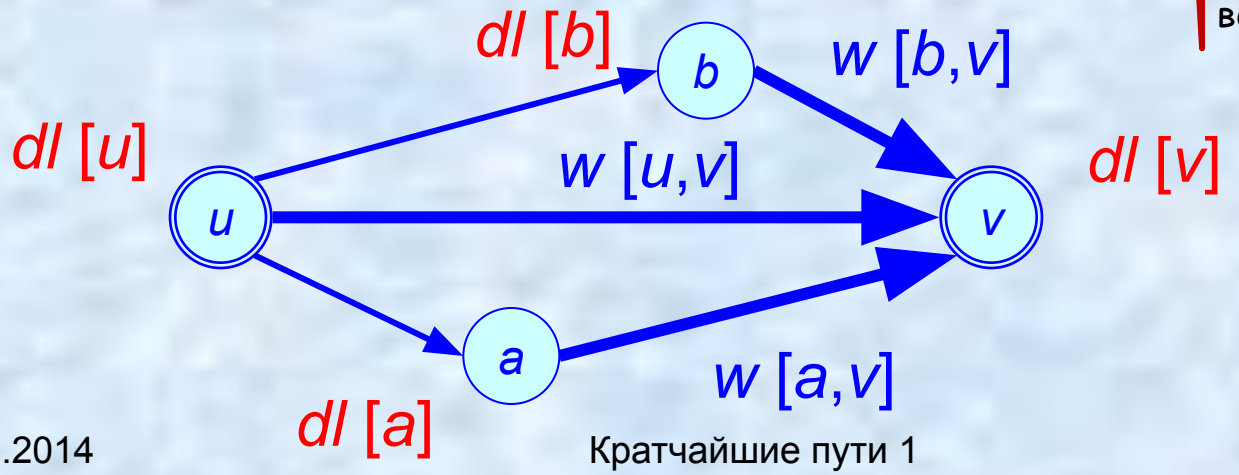
Релаксация (ослабление) ребра $u \to v$

Relax (u, v);




```
for (∀u ∈ Adj_In[v]) Relax(u, v);
```

Релаксация входящих ребер относительно вершины *v*



1

Задачи вычисления расстояний от фиксированной вершины

Релаксация входящих ребер относительно вершины v :

for ($\forall$ u $\in$ Adj_In[v]) Relax(u, v);

В результате

$dl\,[v] = \min\,(\,dl\,[u] + W\,[\,u, v\,]\;\mid\;\forall\, u \in Adj\_In[v]\,)$

Инициализация: $dl\,[v] = W\,[\,s, v\,]$ или $dl\,[v] = \infty$, $dl\,[s] = 0$.

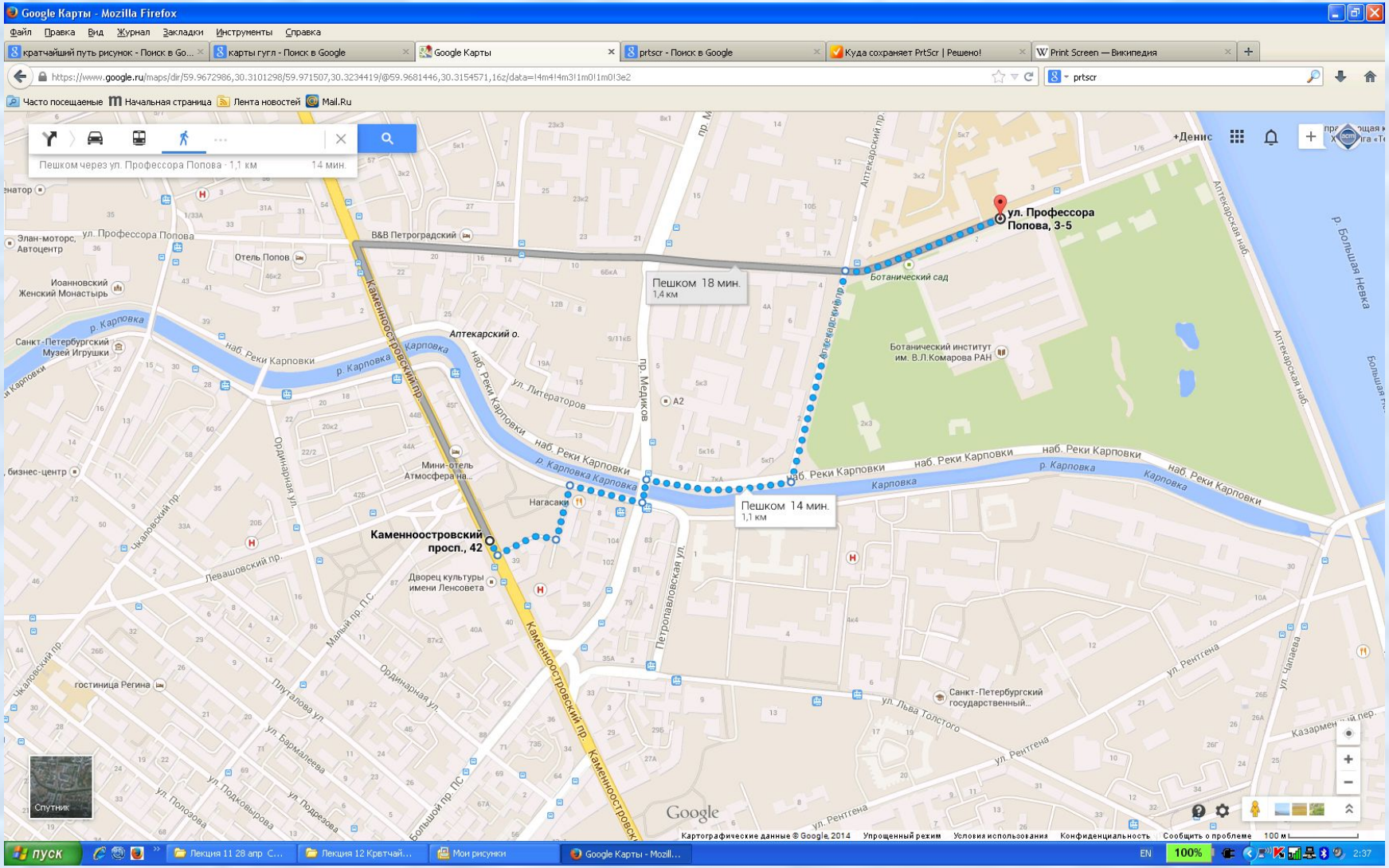
Утверждение 1.

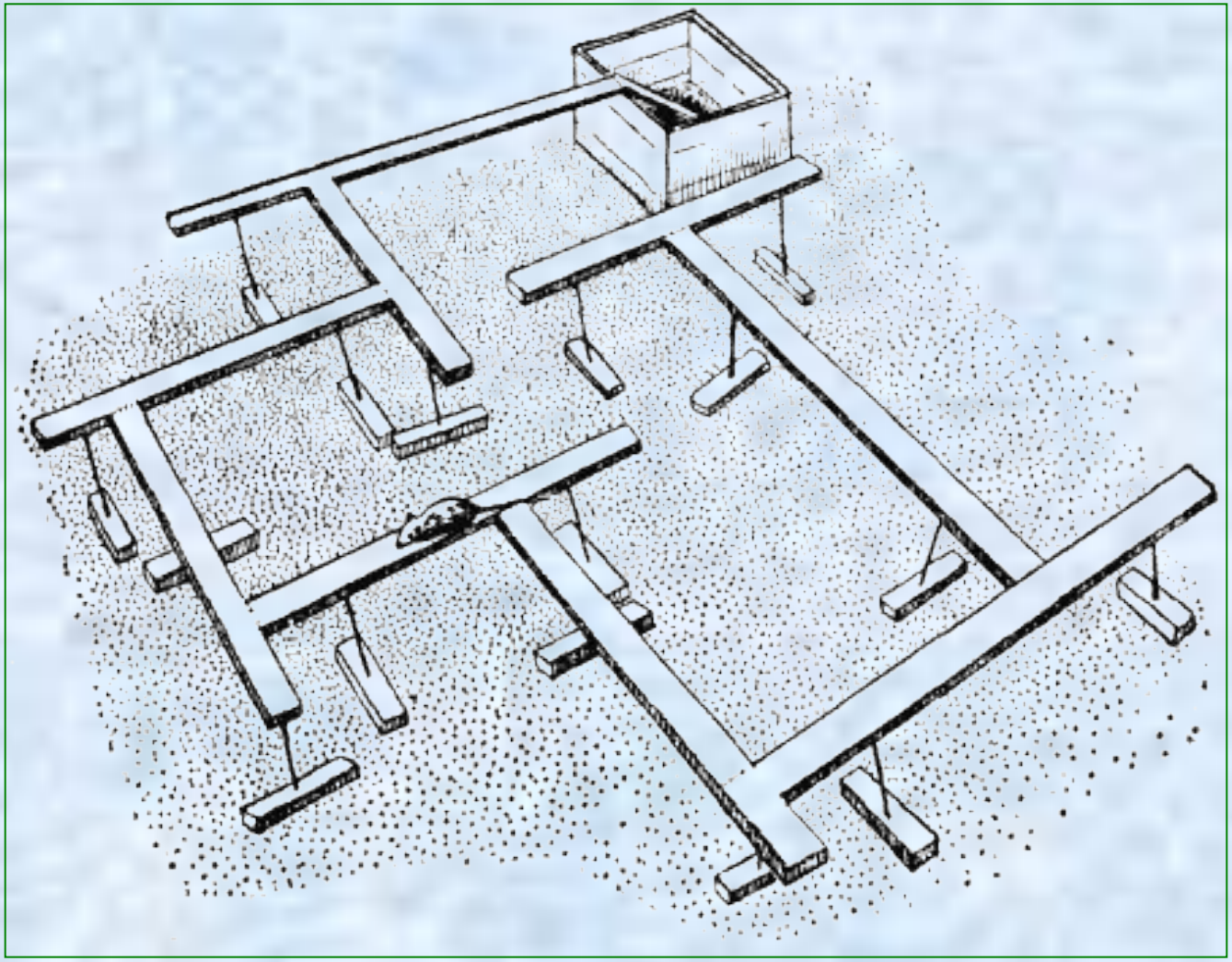
Если $dl\,[v] = d\,(s, v)$, то релаксация не изменит $dl\,[v]$.

Утверждение 2.

Если u лежит на кратчайшем пути $s \Rightarrow v$ и $dl\,[u] = d\,(s, u)$, то после Relax(u, v) будет $dl\,[v] = d\,(s, v)$.

Продолжение на лекции 5 мая





Алгоритм Дейкстры
(Dijkstra E.W. - 1959)

Алгоритм Дейкстры

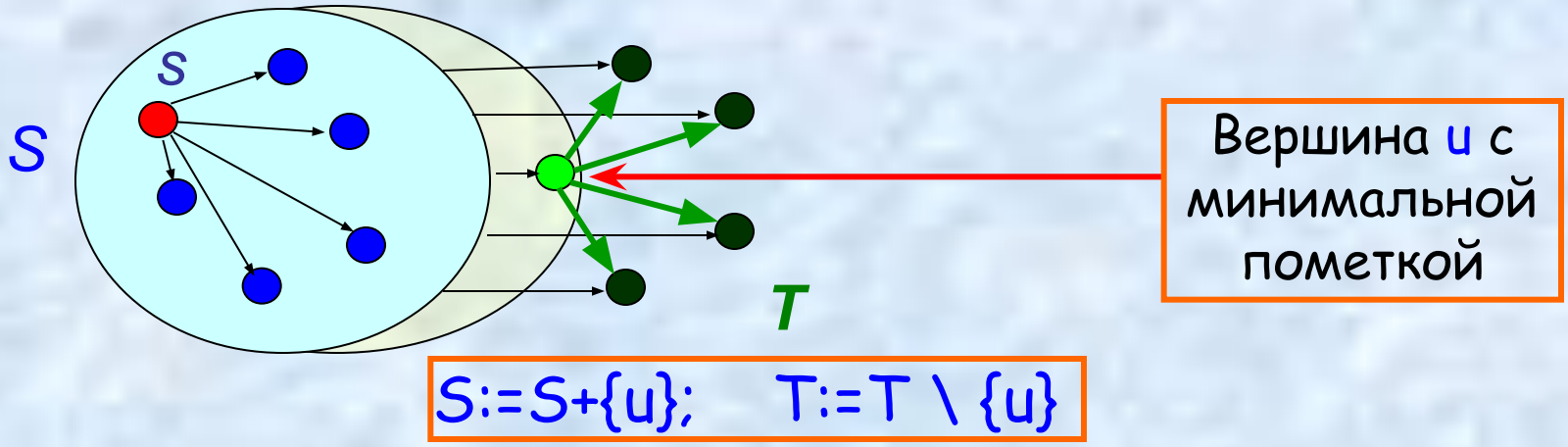
$W[*,*] \geq 0$

Идея алгоритма: $V = S + T$, $S \cap T = \varnothing$

Инвариант:

($\forall\ v \in S$: $DL[v] = d(s,v)$) &

($\forall\ v \in T$: $DL[v]$ = длина кратчайшего из тех путей из s в v, в которых все вершины, кроме v, принадлежат множеству S)



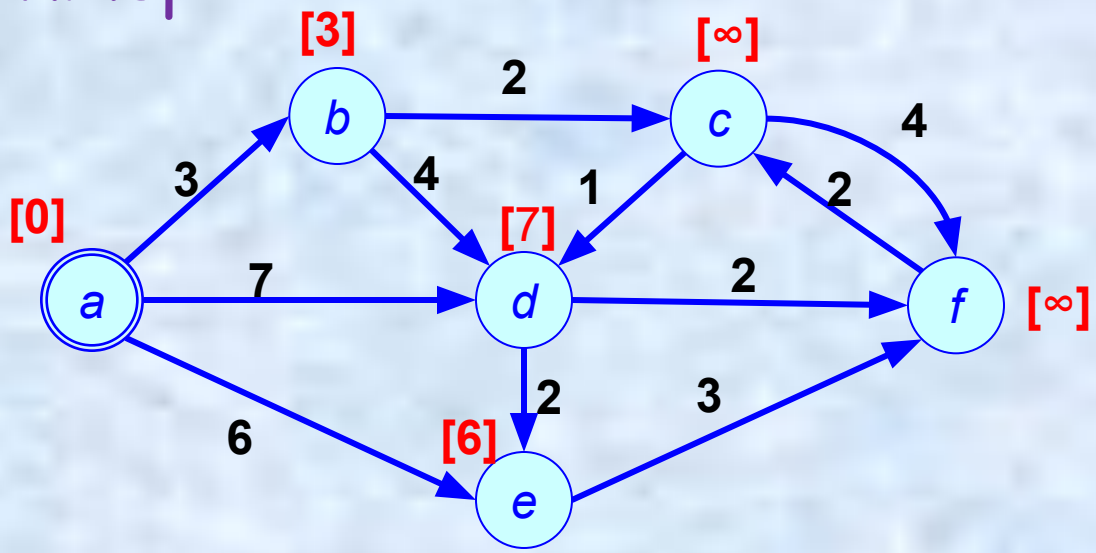
Алгоритм Дейкстры

Дано: Орграф $G=\langle V,E\rangle$ с выделенным источником $s \in V$;
$W[u, v]$ – матрица весов дуг, $u, v \in V$; $W[u, v] \geq 0$
Результат Расстояния от источника s до всех вершин графа $DL[v]=d(s,v)$, $v \in V$;

```
for (∀ v ∈V) DL[v] =W[s,v];
DL[s] =0;
 T = V \ {s};      // S = {{s}}
 while (T ≠ ∅) {
   u = вершина x∈T, такая, что DL[x] = min { DL[p] : p ∈T };
   T =T \ {u};     // S =S+{u}
    for (∀ v ∈T) Relax (u,v);
 } //while
```

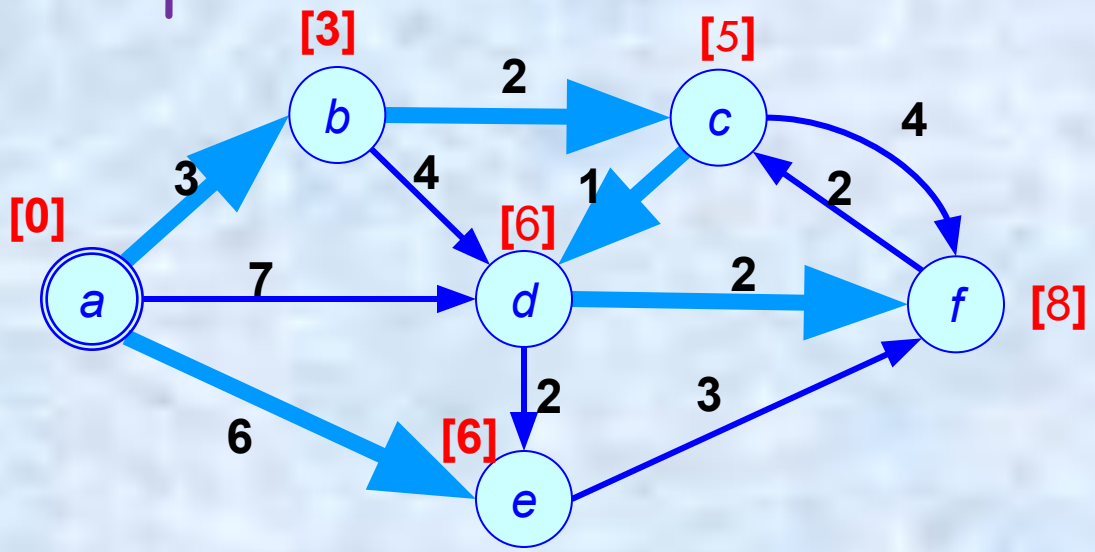
DL[v] = min { DL[v], DL[u]+W[u,v] }

Пример



Шаг	S	DL[b]	DL[c]	DL[d]	DL[e]	DL[f]
0	{a}	3	∞	7	6	∞
1	{a, b}	3	5	7	6	∞
2	{a, b, c}	3	5	6	6	9
3	{a, b, c, e}	3	5	6	6	9
4	{a, b, c, e, d}	3	5	6	6	8
5	{a, b, c, e, d, f}	3	5	6	6	8

Пример

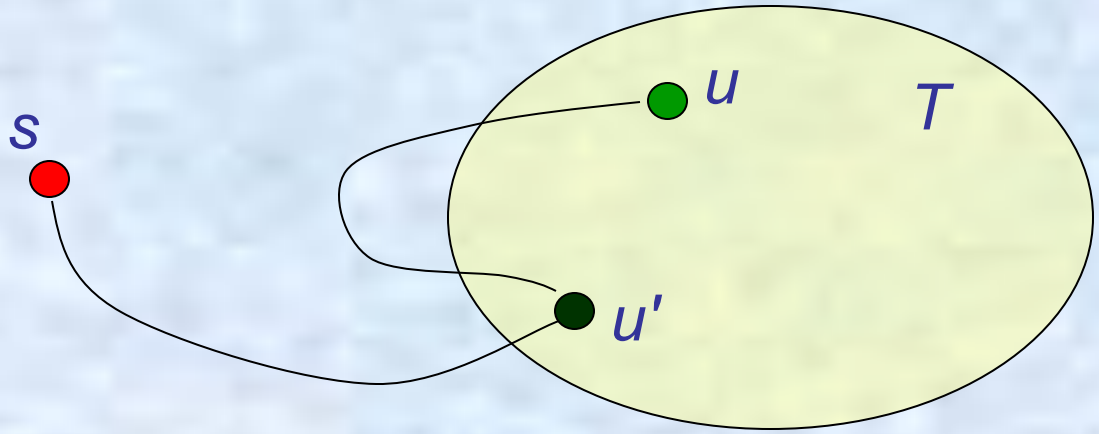


Шаг	S	DL[b]	DL[c]	DL[d]	DL[e]	DL[f]
0	{a}	3	∞	7	6	∞
1	{a, b}	3	5	7	6	∞
2	{a, b, c}	3	5	6	6	9
3	{a, b, c, e}	3	5	6	6	9
4	{a, b, c, e, d}	3	5	6	6	8
5	{a, b, c, e, d, f}	3	5	6	6	8

Алгоритм Дейкстры

Корректность алгоритма: см. инвариант цикла **while;**

от противного



Пусть $u = \arg\min \{ DL[p] : p \in T \}$; но $DL[u] > d(s,u)$.

Тогда $\exists\ u' \in T : DL[u'] = d(s,u') \le d(s,u) < DL[u]$!

(и u' - первая вершина из T на пути $s \Rightarrow u$)

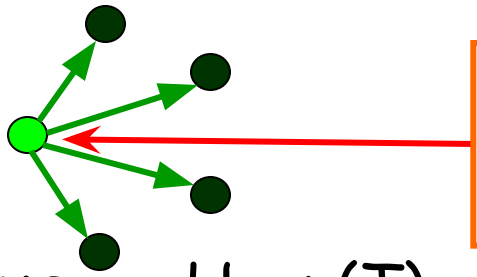
(противоречие) Сложность $O(n^2)$

Алгоритм Дейкстры (модификация O(m log n))

1) Множество T представляется пирамидой Heap(T) с приоритетами DL[v];
при этом **Inv** Heap(T): $(\forall u \in T: (u=\text{отец}(v)) \rightarrow (DL[u] \le DL[v]))$.
Тогда min(T) – в корне Heap(T), и исключение u из T есть удаление корня Heap(T) с восстановлением пирамидальности.

2) Используются списки смежности AdjOut[*].
Тогда обновление DL[v] после выбора вершины u реализуется следующим образом:

```
for (∀v∈ AdjOut[u])
  if (DL[u] + W[u,v] < DL[v]) {
     DL[v] = DL[u]+W[u,v];
     Восстановить пирамидальность Heap(T) вверх от узла v;
  }
```



Каждая дуга графа анализируется один раз (!) и может вызвать действие O(log n)

Следующий

Алгоритм Форда-Беллмана

Алгоритм Форда-Беллмана

нахождения расстояний от источника до остальных вершин в графе, **не содержащем контуров отрицательной длины**

Дано:

Орграф $G=\langle V,E\rangle$ с выделенным источником $s \in V$; $|V| = n$;

$W[u, v]$ – матрица весов дуг, $u, v \in V$;

(граф не содержит *контуров отрицательной длины*)

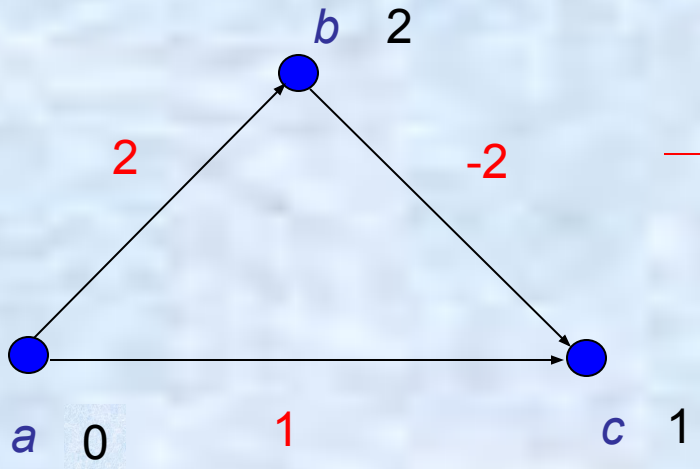
Результат:

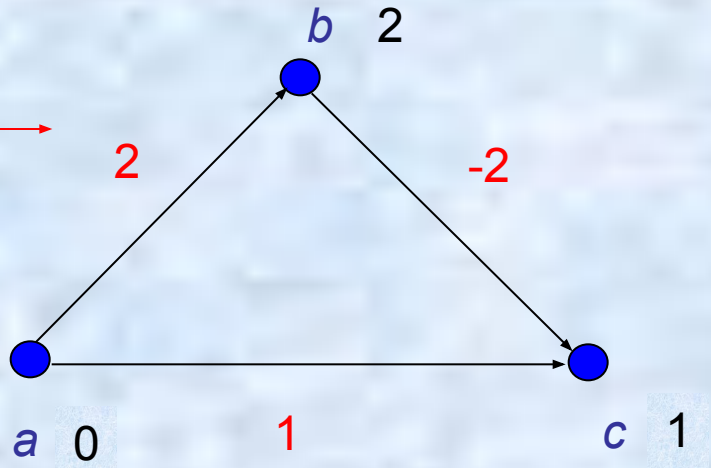
Расстояния от источника s до всех вершин графа

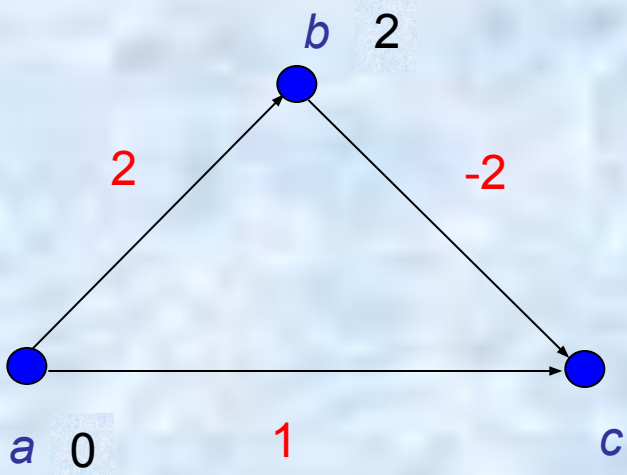
$DL[v] = d(s,v)$, $v \in V$;

Алгоритм Форда-Беллмана

Пример непригодности алгоритма Дейкстры при произвольных весах







Но Relax (b, c) дает DL[c] = 0 !!!

Алгоритм Форда-Беллмана

```
for (∀ v ∈ V) DL[v] =W[s,v];
DL[s] = 0;
for (k=1; k<=(n – 2); k++)
 for (∀ v ∈ V \ {s})
    for (∀ u ∈ V) Relax (u, v);
```

$$DL[v] = \min \{ DL[v],\ DL[u]+W[u,v] \}$$

Сложность $O(n^3)$

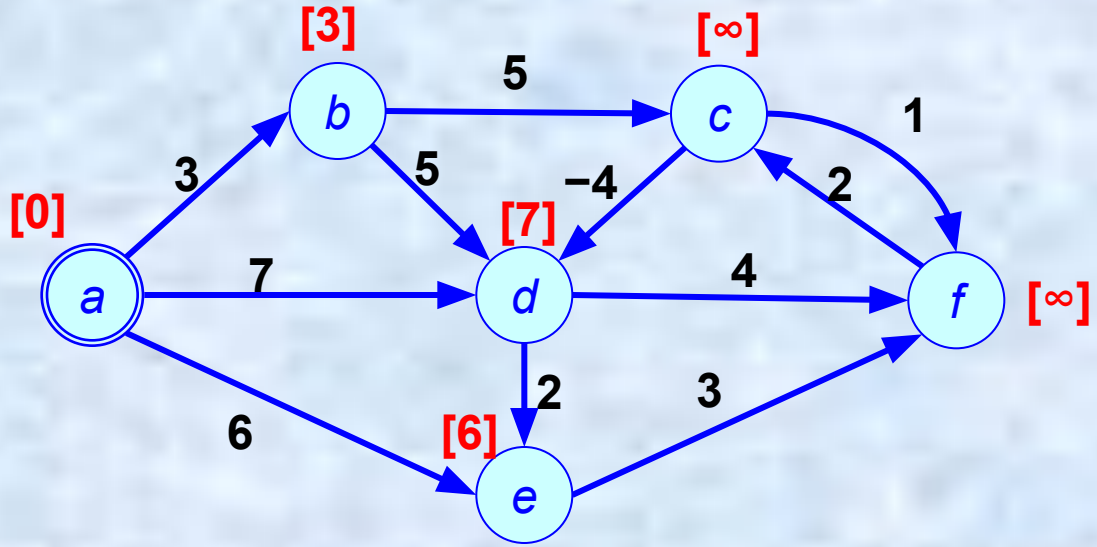
Алгоритм Форда-Беллмана

Вариант сложности $O(nm)$

```
for (∀ v ∈ V) DL[v]=W[s,v];
DL[s] =0;
for (k=1; k<=(n – 2); k++)
    for (∀ e ∈ E) Relax (u, v); //e=(u, v)
```

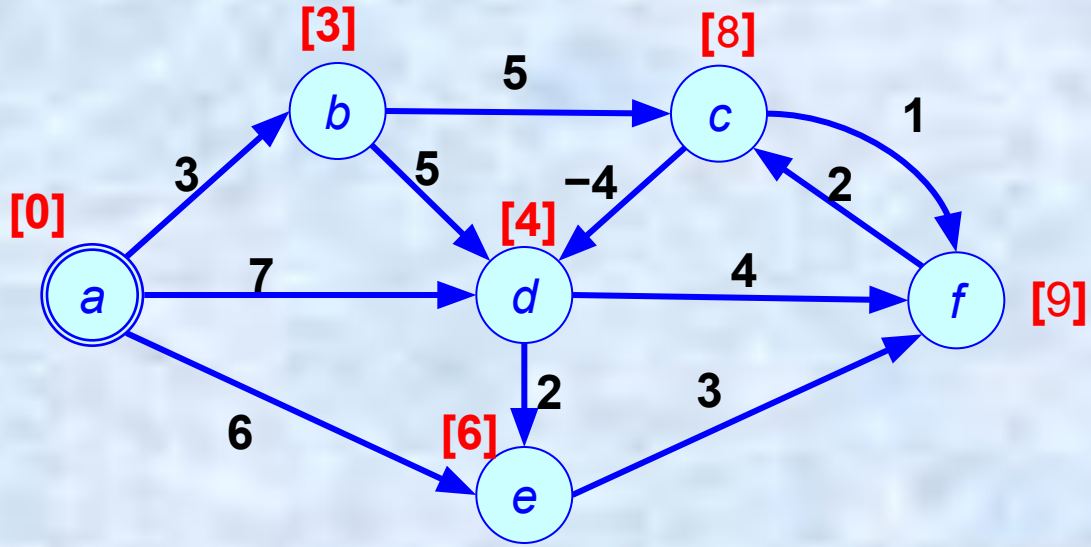
Примечание: если ещё раз (дополнительно) выполнить тело цикла "**for** (k=1; …)" и проверить уменьшение в Relax, то можно обнаружить в графе цикл отрицательной длины (в обоих вариантах)

Пример. n = 6 Шаг 0



Шаг	*DL*[*b*]	*DL*[*c*]	*DL*[*d*]	*DL*[*e*]	*DL*[*f*]
0	3	∞	7	6	∞
1					
2					
3					
4					

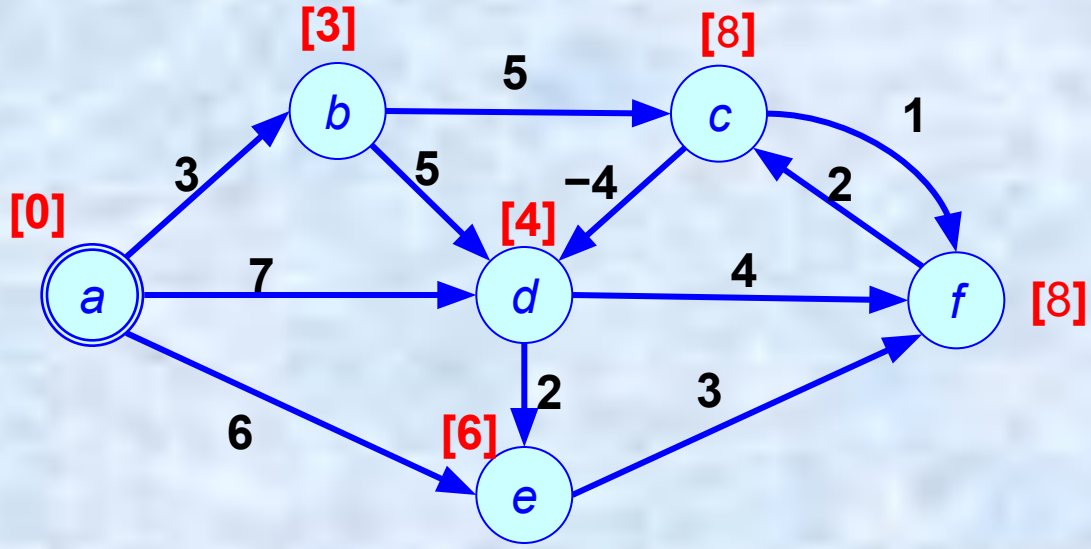
Пример. n = 6 Шаг 1



Шаг	*DL*[*b*]	*DL*[*c*]	*DL*[*d*]	*DL*[*e*]	*DL*[*f*]
0	3	∞	7	6	∞
1	3	8	4	6	9
2					
3					
4					

(a,b)
(a,d)
(a,e)
(b,c)
(b,d)
(d,e)
(d,f)
(e,f)*
(c,d)
(c,f)
(f,c)

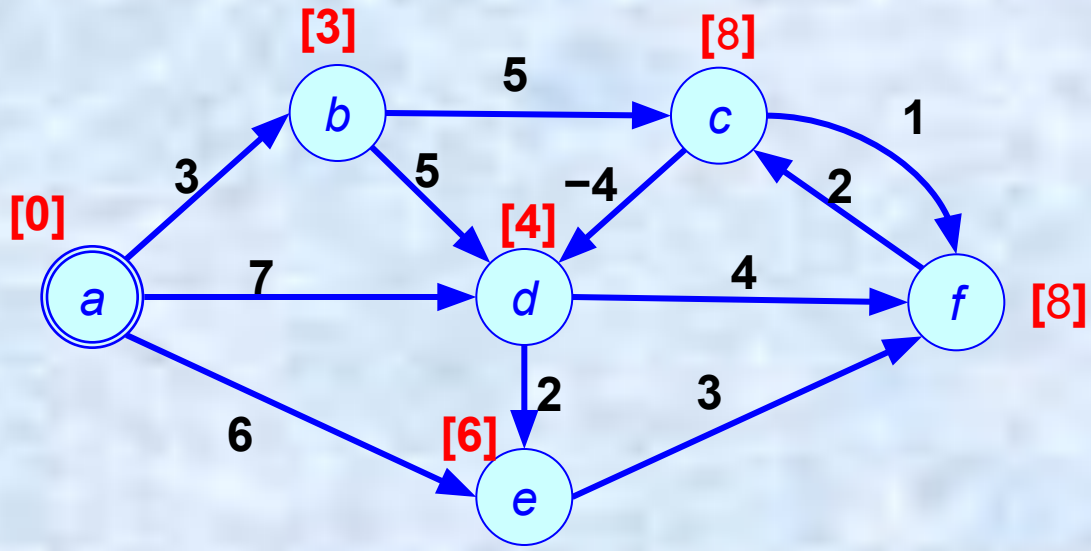
Пример. n = 6 Шаг 2



Шаг	*DL*[*b*]	*DL*[*c*]	*DL*[*d*]	*DL*[*e*]	*DL*[*f*]
0	3	∞	7	6	∞
1	3	8	4	6	9
2	3	8	4	6	8
3					
4					

(a,b)
(a,d)
(a,e)
(b,c)
(b,d)
(d,e)
(d,f)
(e,f)
(c,d)
(c,f)
(f,c)

Пример. n = 6 Шаг 3



Шаг	*DL*[*b*]	*DL*[*c*]	*DL*[*d*]	*DL*[*e*]	*DL*[*f*]
0	3	∞	7	6	∞
1	3	8	4	6	9
2	3	8	4	6	8
3	3	8	4	6	8
4					

(a,b)
(a,d)
(a,e)
(b,c)
(b,d)
(d,e)
(d,f)
(e,f)
(c,d)
(c,f)
(f,c)

Алгоритм Форда-Беллмана

Индуктивное утверждение ("инвариант") цикла

for (k=1; k<=(n – 2); k++) {…} :

P([1..k)) ≡ (d(s,v) ≤ DL[v] ≤ d[k | v]) - *инвариант точки входа* тела цикла,

P([1..k]) ≡ (d(s,v) ≤ DL[v] ≤ d[k+1| v]) - *инвариант точки выхода* тела цикла,

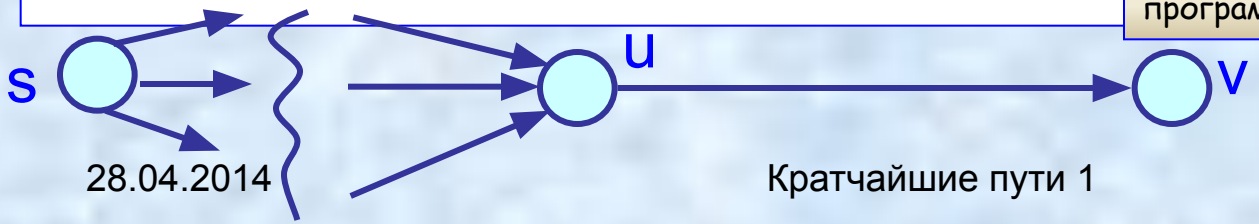
где d[k | v] – длина кратчайшего из путей из s в v, содержащих *не более* k дуг.

Сами величины d[k|v] в алгоритме не используются!
Они нужны только для записи утверждений.

Рекуррентное соотношение:

d[k+1| v] = **min** {d[k | u] + W[u,v] | u ∈ V }

Типичное соотношение динамического программирования



Корректность алгоритма Форда-Беллмана

1. Если P([1..k)) и P([1..k]) действительно *инварианты*, то для остановки необходимо, чтобы после k итераций было бы k + 1 = n – 1 (т. к. путь без циклов в графе с n вершинами может иметь не более n – 1 дуг).

   Отсюда k = n – 2
2. Покажем, что действительно тело внешнего цикла переводит P([1..k)) в P([1..k])

Корректность алгоритма Форда-Беллмана

После очередной итерации

новая

$d(s,v) \le DL[v] \le$ { по алгоритму, с учетом того, что пометки $DL[u]$ могли уже уменьшиться на предыдущих шагах циклов внутри одной итерации по k }

$$\le \min \{DL[u] + W[u,v] \mid u \in V\} \le$$

$$\le \min \{d[k \mid u] + W[u,v] \mid u \in V\} = d[k+1 \mid u]$$

старая

по предположению индукции

Кратчайшие пути между всеми парами вершин

См. лекцию 12

Примечание

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ