ЛЕКЦИЯ 12. АЛГОРИТМЫ НА ВЗВЕШЕННЫХ ГРАФАХ

ВЗВЕШЕННЫЕ ГРАФЫ

О п р е д е л е н и е . Граф называется **взвешенным** (**нагруженным**) если каждому ребру приписано некоторое число – *вес ребра*.

Эти числа составляют весовую матрицу. Веса обычно положительные, но иногда бывают отрицательные (например, в экономике: + прибыль, – убытки).

Если ребра нет, то его **вес удобно считать равным бесконечности**.

О п р е д е л е н и е . Длиной пути на взвешенном графе называется сумма весов составляющих его рёбер. Расстояние между вершинами = длина кратчайшего пути.

З а д а ч а : найти кратчайший путь между вершинами s и t.

Заметим, что если найти расстояния от s, до всех остальных, то путь легко «раскрутить» в обратную сторону: если расстояние от s до t равно d[t], то найдётся такая вершина v, что

$$d[t] = d[v] + A[v,t]$$

(v – предпоследняя вершина на кратчайшем пути). Такую v легко найти, перебрав все вершины. Затем находим вершину, предшествующую v, и т.д., пока не вернёмся в s. Но можно обойтись и без перебора.

АЛГОРИТМ ФОРДА-БЕЛЛМАНА (1969)

Допускает отрицательные веса. Будем считать, что в графе нет циклов с отрицательным весом. Граф может быть как ориентированным, так и не ориентированным. Из этих условий следует, что кратчайший путь не может содержать циклов – т.к. они имеют положительный вес, то их можно выбросить и длина пути от этого уменьшится.

ОПИСАНИЕ АЛГОРИТМА

Каждой вершине сопоставим оценку d[v] — минимальное известное к этому моменту расстояние от *s* до этой вершины. На каждом шаге будем пытаться улучшить каждую из оценок, считая предпоследней вершиной пути по очереди каждую из вершин 1,2, …, N.

ПРОГРАММА

```
d = A[source-1].copy()
for k in range(n-2):
    for v in range(n):
        for u in range(n):
            d[v] = min(d[v],d[u]+A[u][v])
```

СЛОЖНОСТЬ

Сложность = $O(p^3)$

ДОКАЗАТЕЛЬСТВО

Обозначим через $d^{(k)}(v)$ – длину кратчайшего пути между s и v, *содержащего не более k рёбер*.

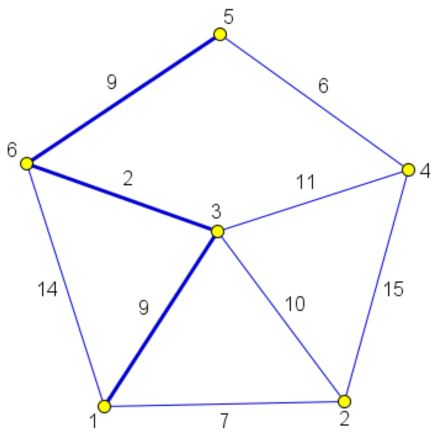
Очевидно, что $d^{(k+1)}(v)=\min(d^{(k)}(u)+A[u,v])$, где минимум берётся по всем u.

Перед началом цикла по параметру k в массиве d находятся $A[s,v]=d^{(1)}(v)$;

После первого прохода дам будут $d^{(2)}(v)$, второго - $d^{(3)}(v)$, …, после (N-2)-го – $d^{(N-1)}(v)$.

Но ведь $d^{(N-1)}(v)$ это и есть длины кратчайших путей, т.к. кратчайший путь не может содержать больше (N-1) вершины: тогда в нём есть повторяющиеся вершины, т.е. циклы, чего быть не может.

П р и м е р . Найти кратчайший путь из 1 в 5:



	Вершины						
	1	2	3	4	5	6	
Шаги k	0	7	9	∞	∞	14	
1	0	7	9	20	23	11	
2	0	7	9	20	20	11	
3	0	7	9	20	20	11	OK
4							

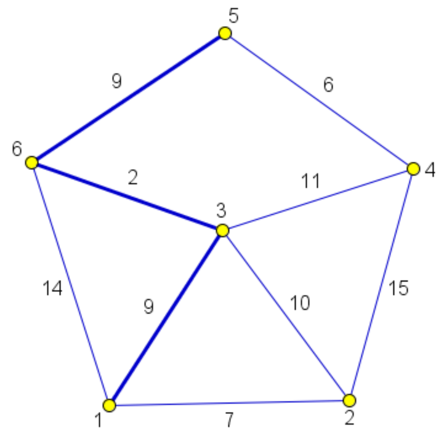
Длина пути = 20

Путь: 5 ← 6 ← 3 ← 1

АЛГОРИТМ ДЕЙКСТРЫ (1959)

Будем считать, что веса всех рёбер неотрицательные. Граф может быть как ориентированным, так и не ориентированным.

П р и м е р . Найти кратчайший путь из 1 в 5:



Шаги	Вершины						Min
	1	2	3	4	5	6	
1	0	∞	∞	∞	∞	∞	1
2		7	9	∞	∞	14	2
3			9	22	∞	14	3
4				20	∞	11	6
5				20	20		4
					20		5

Длина пути = 20

Путь: 5 ← 6 ← 3 ← 1

ОПИСАНИЕ АЛГОРИТМА

Каждой вершине сопоставим оценку d[v] — минимальное известное к этому моменту расстояние от *s* до этой вершины. На каждом шаге одна из оценок (а именно, минимальная) будет считаться окончательной (т.е. равной расстоянию), а остальные будут через неё улучшаться.

Инициализация. Оценка самой вершины *s* полагается равной 0, оценки остальных вершин — бесконечности. Это отражает то, что расстояния от *s* до других вершин пока неизвестны. Все вершины графа считаются непомеченными.

Шаг алгоритма. Если все вершины помечены, алгоритм завершается. В противном случае, из ещё не помеченных вершин выбирается вершина u, имеющая минимальную оценку d[u] и она помечается. Рассматриваются всевозможные пути, в которых u является **предпоследним** пунктом. Для каждого непомеченного соседа v вершины u попытаемся улучшить оценку d[v] через d[u]+A[u,v].

После повторения шага (p-1) раз все оценки d[v] будут равны рассояниям от s до v.

ПРОГРАММА

```
# В начале все вершины не помечены, а все расстояния равны бесконечности
Mark = [False]*n
d = [inf]*n
d[source-1] = 0
prev = [None]*n
# Находим расстояния
```

```
for i in range(n-1):
    m = inf
    for v in range(n):
        if (not Mark[v]) and (d[v]<m):
            m = d[v]
            u = v
    Mark[u] = True
    for v in range(n):
        if (not Mark[v]) and (d[u]+A[u][v]<d[v]):
            d[v] = d[u] + A[u][v]
            prev[v] = u
#print(d)
# Раскручиваем путь по масиву Prev
Path = [target-1]
v = target-1
while v!=source-1:
    v = prev[v]
    Path.insert(0,v)
print("Алгоритм Дейкстры:")
printPath(Path)
print("Расстояние =",d[target-1])
```

СЛОЖНОСТЬ

Сложность = $O(p^2+q)=O(p^2)$

ДОКАЗАТЕЛЬСТВО

Инвариант цикла:

1) для каждой помеченной вершины её оценка равна длине кратчайшего пути;
2) для каждой непомеченной вершины её оценка равна длине кратчайшего из всех путей, у которых предпоследняя вершина помечена.

Пункт 2 выполнен по построению оценок (мы каждый раз переоцениваем все оценки именно таким образом). Пусть пункт 1 **в первый раз** в какой-то момент не выполнен, то есть

$$\text{dist}(s,v) < d[v]$$

для вершины v с минимальной оценкой. Рассмотрим абсолютно кратчайший путь до **v**: его предпоследняя вершина **u** в этот момент не помечена (иначе нарушился бы пункт 2). Рассмотрим первую непомеченную вершину **w** на этом пути, которая идёт после серии помеченных (такая есть, поскольку начальная вершина помечена). Но тогда для этого момента имеем:

$$d[w] = \text{dist}(s,w) <= \text{dist}(s,v) < d[v],$$

что противоречит выбору вершины v, как вершины с минимальной оценкой. Но если 1) - это инвариант, то как только вершина t станет помеченной, её оценка даст длину кратчайшего пути: d[t]=dist(s,t).

АЛГОРИТМ ФЛОЙДА-УОРШАЛЛА (1962)

Если нужно найти расстояния между всеми вершинами графа, то применение алгоритма Дейкстры потребует $O(p^3)$ шагов. Есть алгоритм, который имеет ту же сложность, но делает это проще и быстрее. Для применения достаточно, чтобы в графе **не было циклов отрицательной длины**.

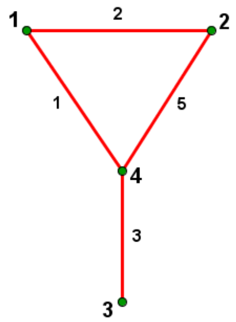
ПРОГРАММА

```
for k in range(n):
    for u in range(n):
        for v in range(n):
            A[u][v] = min(A[u][v], A[u][k]+A[k][v])
```

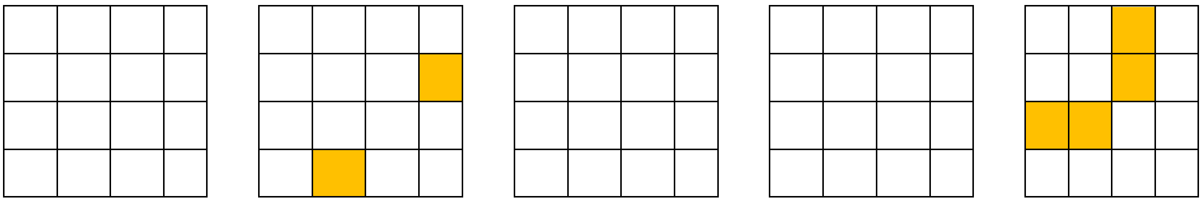
СЛОЖНОСТЬ

Сложность = $O(p^3)$

П р и м е р.



A: k=1: k=2: k=3: k=4:



ДОКАЗАТЕЛЬСТВО

Обозначим через $d^{(k)}(u,v)$ – длину кратчайшего пути между u и v, если в качестве промежуточных разрешается использовать только вершины из множества 1..k. Назовём этот путь k-кратчайшим путём.

Очевидно, что $d^{(0)}(u,v)$ – вес ребра.

Существует два варианта для нахождения $d^{(k)}(u,v)$:

- если k-кратчайший путь из u в v не проходит через вершину k, то $d^{(k)}(u,v)= d^{(k-1)}(u,v)$;
- если он не проходит через вершину k, то $d^{(k)}(u,v)= d^{(k-1)}(u,k)+ d^{(k-1)}(k,v)$;

Остаётся проделать N итераций, чтобы получить $d^{(N)}(u,v)$ – то есть, расстояния.