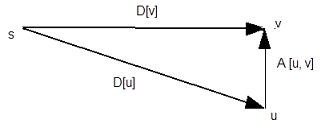
**14. АЛГОРИТМ ФОРДА – БЕЛЛМАНА**

Рассмотрим теперь алгоритм, формирующий вектор расстояний *D* для самого общего случая, т.е. для орграфа с вещественными весами дуг и без контуров отрицательного веса.

Результат работы алгоритма – вектор расстояний D[*v*] от фиксированного источника *s* до всех остальных вершин графа. За начальные (приближенные) значения D[*v*] возьмем веса дуг *A*[*s*, *v*]. Они будут соответствовать путям из *s* в *v* длиной в одну дугу. Если дуги (*s*→*v*) не существует, то начальное D[*v*] принимается равным +∞. Расстояние D[*s*] от источника до самого себя всегда считается равным 0.

Каким образом можно уточнить значения D[*v*]? Взять все пути из *s* в *v* длиной в две дуги и сравнить их с начальными D[*v*]. За окончательное значение D[*v*] принимается минимальное. Затем берутся пути длиной в три дуги и т.д. Процесс пересчета будет продолжаться до тех пор, пока не будут рассмотрены пути длиной в n-1 дугу. Нет смысла рассматривать пути длиной в *n* и более дуг, так как они будут содержать контур неотрицательного веса, который только увеличит суммарную длину пути.

Остается выяснить, каким образом из пути в *k* дуг можно получить все пути длиной в k + 1 дугу. Из рис. 14.1 видно, что, добавив между *s* и *v* промежуточную (предпоследнюю) вершину *u*, мы автоматически удлиняем путь ровно на одну дугу. Длина старого пути равна *D*[*v*], длина нового пути равна *D*[*u*] + *A*[*u*, *v*].

  
Рис. 14.1. Удлинение пути на одну дугу

**Алгоритм Форда – Беллмана**

Данные: Орграф *G* = <*V*, *E*> с *n* вершинами и выделенным источником *s*, заданный списками инцидентности PRED, матрица весов A[*u*, *v*], *u*,*v* ∈ *V*; граф не содержит контуров отрицательной длины.

Результаты: Расстояния D[*v*] от источника *s* до всех остальных *v*∈*V*.

1 begin

2 for v∈V do D[v]:= A[s,v];

3 D[s]:= 0;

4 for k := 1 to n – 2 do

5 for v∈V–[s] do

6 for u∈PRED[v] do

7 D[v]:= min(D[v], D[u]+A[u,v]);

8 end.

***Вопрос.*** Пусть путь от источника до вершины *v* не существует. Может ли быть (в результате работы алгоритма) сформировано расстояние D[*v*] меньшее +∞?

Возможно, для вычисления D[*v*] нам не потребуются все n-2 итерации. Момент, когда D[*v*] будет вычислено окончательно, легко определить. Это произойдет, когда выполнение цикла 4 не вызовет изменения ни одного значения D[*v*].

**Вычислительная сложность алгоритма**

Вычислительная сложность алгоритма *О*(*nm*). Если граф является «разреженным» и число дуг *m* ≈ *n* , то можно считать вычислительную сложность равной *О*(*n*2), иначе *О*(*n*3).

**Трассировка работы алгоритма Форда – Беллмана (рис. 13.1):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | D[1] | D[2] | D[3] | D[4] | D[5] |
|  | 0 | 1 | +∞ | +∞ | 3 |
| 1 | 0 | 1 | 4 | 4 | 9 |
| 2 | 0 | 1 | 4 | 3 | –1 |
| 3 | 0 | 1 | 4 | 3 | –1 |

**Алгоритм Форда – Беллмана переносится на неориентированные графы только с неотрицательными весами.** Если же у графа есть хотя бы одно ребро отрицательного веса, то при расщеплении на пару противоположно направленных дуг оно даёт контур отрицательного веса.

***Ответ*.** Да, может. В формула пересчёта расстояния D[*v*]:= *min*{D[*v*], D[*u*]+A[*u*, *v*]} может произойти сложение +∞ с отрицательным числом, например, путь до u не существует, а дуга (*u*→*v*) имеет отрицательный вес. Поэтому в формуле пересчёта сложение производится только в случае, если и D[*u*], и A[*u*, *v*] меньше +∞.