Кратчайшие пути в графе

Постановка задачи. Вывод пути

Дан ориентированный граф $G=(V,E)$, веса дуг – $A[i,j]$ ($i,j=1..n$, где n – количество вершин графа), начальная и конечная вершины – $s, t \in V$. Веса дуг записаны в матрице смежности A, если вершины i и j не связаны дугой, то $A[i,j]=\infty$. Путь между s и t описывается как последовательность вершин $v_0 = s, v_1, v_2, \ldots v_{q-1}, v_q=t$ и оценивается $\sum_{i=1}^{q} A[v_{i-1}, v_i]$. Требуется найти путь с минимальной оценкой.

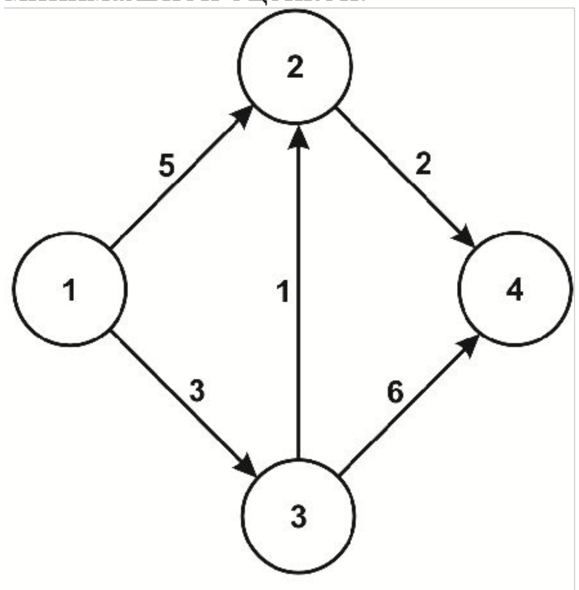

Рис. 12.1. Пример графа, для иллюстрации понятия «кратчайший путь»

Примечание. При реализации алгоритмов на компьютере требуется моделировать бесконечность (символ ∞) путем, например, работы с максимальным значением чисел в используемом типе. Естественно, что в логике появляются дополнительные проверки.

Пример. На рис. 12.1 приведен пример простейшего графа. Кратчайший путь из первой вершины в четвертую проходит через третью и вторую вершины и имеет оценку 6.

Пример. На рис. 12.2 приведен пример графа. Он имеет простой цикл $3 \to 4 \to 2 \to 3$ с отрицательным суммарным весом. При поиске кратчайшего пути между первой и пятой вершинами, обходя этот цикл достаточное количество раз, можно получить оценку, меньшую любого целого числа.

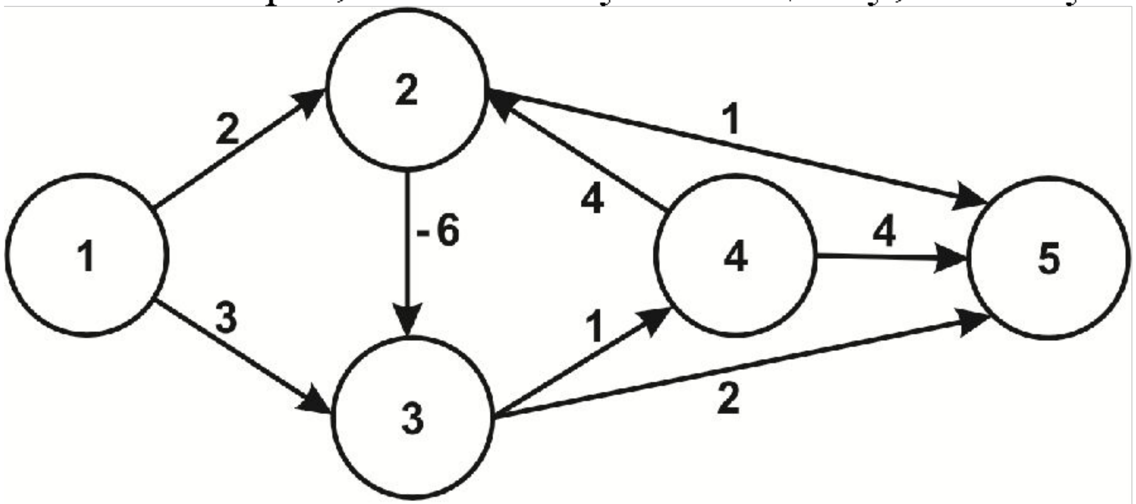

Рис. 12.2. Пример графа с отрицательным циклом

Ограничимся рассмотрением графов без циклов с отрицательным суммарным весом. Поиск в графе таких циклов – отдельная задача.

Нам необходимо найти кратчайший путь, то есть путь с минимальным весом, между двумя вершинами графа. Эта задача логически разбивается на две подзадачи: нахождение минимальной оценки пути и вывод пути (при известной оценке). Неизвестны алгоритмы, определяющие только оценку между заданной парой вершин, все они определяют оценки от вершины *s* до всех остальных вершин графа. Определим *D* *(D:Array*[*1..n*] *Of Integer)*. Предположим, что мы определили значения элементов массива *D* – решили первую подзадачу. Найдем сам кратчайший путь. Для *s* и *t* существует такая вершина *v*, что *D*[*t*]=*D*[*v*]+*A*[*v,t*]. Запомним *v* (например, в стеке). Повторим процесс поиска вершины *u*, такой, что *D*[*v*]=*D*[*u*]+*A*[*u,v*], и так до тех пор, пока не дойдем до вершины с номером *s*. Последовательность *t, v, u, ..., s* дает кратчайший путь. Реализация этой логики имеет вид:

```
Procedure Way(q:Integer); {D, A, s, t – глобальные переменные.}
 Var j:Integer;
 Begin
  If q<>s Then Begin
   j:=1;
   While (j<=n) And (D[t] <> D[j] + A [j,t] Do j:=j+1;
   If j<=n Then Way(j);
  End;
  Write (q,'  ');
 End;
```

Алгоритм Л. Форда и Р. Беллмана. Дан ориентированный взвешенный граф G, матрица смежности A – веса дуг, вершина источник s, циклов с отрицательным суммарным весом нет. Результат – массив оценок D.

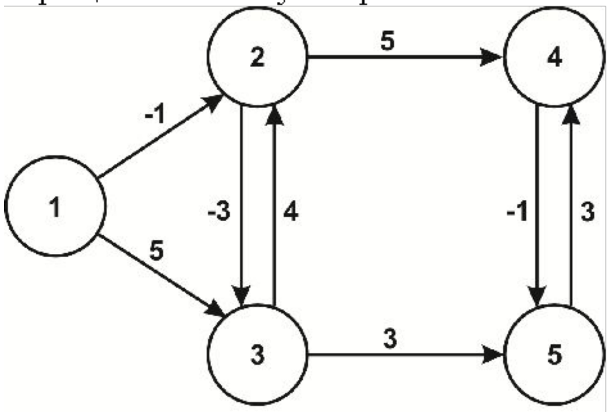

Рис. 12.3. Пример графа для иллюстрации логики алгоритма Форда-Беллмана

Пример. На рис. 12.3 показан граф. Его матрица смежности (весов) имеет вид: $A = \begin{pmatrix} \infty & -1 & 5 & \infty & \infty \\ \infty & \infty & -3 & 5 & \infty \\ \infty & 4 & \infty & \infty & 3 \\ \infty & \infty & \infty & \infty & -1 \\ \infty & \infty & \infty & 3 & \infty \end{pmatrix}$.

Начальное значение массива D (0, ∞, ∞, ∞, ∞). Ищем кратчайшие пути из первой вершины. Оценки до всех вершин не получены. На первой итерации – путь состоит из одной дуги – получаем оценки D (0, -1, 5, ∞, ∞). На второй итерации – путь состоит не более чем из двух дуг – новые оценки D (0, -1, -4, 4, 8). Для вершины с номером 3 оценка улучшается. На третьей итерации улучшается оценка до вершины с номером 5 – D (0, -1, -4, 4, -1). И наконец, на пятой итерации изменяется оценка четвертой вершины – D (0, -1, 4, 2, -1). Дальнейшие действия бессмысленны – кратчайший путь не может содержать более n-1 дуги. Большее количество дуг предполагает существование цикла в пути, а суммарный вес всех циклов положительный.

Приведем логику реализации алгоритма.

```
Procedure Solve;
 Var i,j,t:Integer;
 Begin
  For j:=1 To n Do D[j]:=A[s,j];
  D[s]:=0;
  For i:=1 To n-2 Do {Количество итераций, одна уже выполнена.}
   For j:=1 To n Do {Вершина, до которой пытаемся улучшить оценку.}
    If j <> s Then
     For t:=1 To n Do D[j]:=Min(D[j],D[t]+A[t,j]); {Поиск вершины, из которой можно улучшить оценку. Функция (Min) определения минимального из двух чисел не приводится.}
 End;
```

Примечание. При выполнении операции сложения *D[t]+A[t,j]* может возникнуть недоразумение. Так, если значение ∞ моделируется на компьютере максимальным числом 32767 типа *Integer*, то результат может выйти за пределы диапазона значений этого типа.

Алгоритм Е. Дейкстры (1959 г.). Дан ориентированный граф $G=(V,E)$, s – вершина-источник; матрица смежности A (*A:Array[1..n,1..n] Of Integer*); для любых u, $v \in V$ вес дуги неотрицательный ($A[u,v] \geq 0$). Результат – массив кратчайших расстояний D.

Неотрицательность весов дуг позволяет сделать следующее умозаключения. На какой-то итерации есть оценки расстояний до вершин графа от вершины s. Значение минимального элемента в этом массиве не может быть изменено на последующих итерациях, ибо количество дуг в пути только увеличится, а их веса неотрицательны. Эту вершину можно исключить из дальнейших шагов по формированию оценок. Следовательно, разумно ввести множество вершин T, для которых еще не вычислена оценка расстояние, и на каждой итерации исключать по одной вершине из него и формировать новые оценки до оставшихся вершин только от исключаемой вершины.

Пример. На рис. 12.4 приведен пример графа, его матрица смежности имеет вид: $A = \begin{pmatrix} \infty & 3 & 7 & \infty & \infty & \infty \\ 1 & \infty & 2 & \infty & \infty & 1 \\ \infty & 1 & \infty & 4 & 4 & \infty \\ \infty & \infty & \infty & \infty & 1 & 5 \\ \infty & \infty & 1 & \infty & \infty & 3 \\ \infty & \infty & \infty & 2 & \infty & \infty \end{pmatrix}$ и s=1. Логику работы отразим в табл. 12.1.

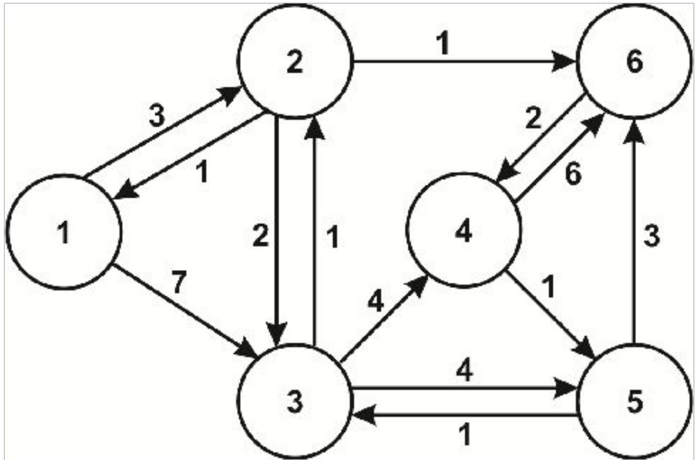

Рис. 12.4. Пример графа для иллюстрации логики работы алгоритма Дейкстры

В последнем столбце показано изменение множества вершин T. Получив на первой итерации массив оценок, мы находим минимальное значение. Оно соответствует вершине с номером 2. Исключаем её из T и на следующей итерации делаем оценки до оставшихся вершин, от вершины с номером 2 (строка 2 табл. 12.1). Очередной минимум достигается на шестой вершине (минимальные элементы на каждой итерации выделены жирным шрифтом).

Таблица 12.1

№ итерации	*D[1]*	*D[2]*	*D[3]*	*D[4]*	*D[5]*	*D[6]*	*T*
1	0	**3**	7	∞	∞	∞	[2,3,4,5,6]
2	0	3	5	∞	∞	**4**	[3,4,5,6]
3	0	3	**5**	6	∞	4	[3,4,5]
4	0	3	5	**6**	9	4	[4,5]
5	0	3	5	6	**7**	4	[5]

Формализованная запись логики имеет вид:

```
Procedure Solve;
  Var  i,u: Integer;
        T:Set Of 1..n;
  Begin
   For i:=1 To n Do D[i]:=A[s,i];
   D[s]:=0;
   T:=[1..n]-[s];
   While T<>[ ] Do Begin
    u:=<значение l, при котором достигается Min_{l∈T}(D[l])>;{Обычная логика поиска индекса минимального элемента в массиве.}
    T:=T-[u];
    For i:=1 To n Do
     If i In T Then D[i]:=Min(D[i],D[u]+A[u,i]);
   End;
  End;
```

Пути в бесконтурном графе. Дан ориентированный граф $G=(V,E)$ без циклов, веса дуг произвольны. Результат – массив кратчайших расстояний (длин) D от фиксированной вершины s до всех остальных. Утверждение – в произвольном бесконтурном графе вершины можно перенумеровать так, что для каждой дуги (i,j) номер вершины i будет меньше номера вершины j. У графа при этом существует по крайней мере одна вершина, в которую заходит нулевое количество дуг. Задача, для её описания, требует следующих структур данных:

- массива *NumIn*, *NumIn[i]* определяет число дуг, входящих в вершину с номером i;
- массива *Num*, *Num[i]* определяет новый номер вершины i;
- массива *St,* для хранения номеров вершин, в которые заходит нулевое количество дуг. Работа с массивом осуществляется по принципу стека;
- переменной *nm*, текущий номер вершины.

Суть алгоритма. Вершина i, имеющая нулевое значение *NumIn*, заносится в *St*, ей присваивается текущее значение *nm* (запоминается в *Num*), и изменяются значения элементов *NumIn* для всех вершин, связанных с i. Процесс продолжается до тех пор, пока *St* не пуст.

Пример. На рис. 12.5 приведен пример графа. Трассировка работы алгоритма для него описана в табл. 12.2, а логика – в процедуре *Change_Num*.

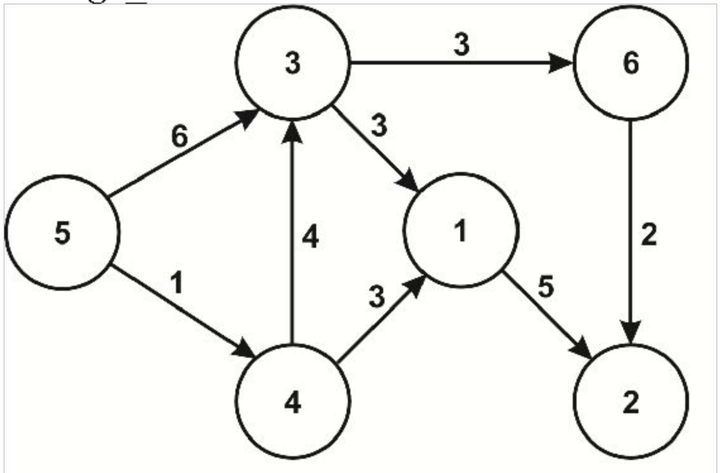

Рис. 12.5. Пример графа для иллюстрации логики поиска кратчайших путей в безконтурном графе

Таблица 12.2

№ итерации	*NumIn*	*Num*	*St*	*Nm*
Начальная	[2,2,2,1,0,1]	[0,0,0,0,0,0]	[5]	0
1	[2,2,1,0,0,1]	[0,0,0,0,1,0]	[4]	1
2	[1,2,0,0,0,1]	[0,0,0,2,1,0]	[3]	2
3	[0,2,0,0,0,0]	[0,0,3,2,1,0]	[6,1]	3
4	[0,1,0,0,0,0]	[0,0,3,2,1,4]	[1]	4
5	[0,0,0,0,0,0]	[5,0,3,2,1,4]	[2]	5
6	[0,0,0,0,0,0]	[5,6,3,2,1,4]	[ ]	6

```
Procedure Change_Num;{A, Num - глобальные структуры данных.}
 Var NumIn,St:Array[1..n] Of Integer;
        i,j,u,nm,yk:Integer;
 Begin
  For i:=1 To n Do Begin NumIn[i]:=0;
                                          Num[i]:=0;
                               End;
  For i:=1 To n Do
   For j:=1 To n Do
    If A[i,j]<>0 Then Inc(NumIn[j]);{Формируем массив NumIn.}
    nm:=0;yk:=0;
    For i:=1 To n Do {Находим вершины с нулевой степенью захода и заносим
их номера в стек.}
     If NumIn[i]=0 Then Begin Inc(yk);
                                              Stack[yk]:=i;
                                      End;
    While yk<>0 Do Begin {Пока стек не пуст.}
     u:=Stack[yk];
     Dec[yk];
     Inc(nm);
     Num[u]:=nm;{Присваиваем вершине из стека очередной номер.}
     For i:=1 To n Do
      If A[u,i]<>0 Then Begin
                                      Dec(NumIn[i]); {Уменьшаем степень захода у вершин,
в которые идут дуги из u.}
                                      If NumIn[i]=0 Then Begin {Если новое значение
степени захода вершины равно нулю, то заносим её номер в стек.}
                                                            Inc(yk);
                                                           Stack[yk]:=i;
                                                          End;
                                    End;
    End;
 End;
```

Кратчайшие пути между всеми парами вершин (алгоритм Р. Флойда, С. Уоршалла). Дан ориентированный граф $G=(V,E)$ с матрицей весов *A(A:Array[1..n,1..n] Of Integer)*. В результате должна быть сформирована матрица D кратчайших расстояний между всеми парами вершин графа и кратчайшие пути.

Идея алгоритма. Обозначим через $D^m[i,j]$ оценку кратчайшего пути из i в j с промежуточными вершинами из множества $[1..m]$. Тогда имеем:

$D^0[i,j]:=A[i,j]$,

$D^1[i,j]:=Min(D^0[i,j],D^0[i,1]+D^0[1,j])$,

$D^2[i,j]:=Min(D^1[i,j],D^1[i,1]+D^1[1,j])$,

…………..

$D^{(m+1)}[i,j]=Min(D^m[i,j],D^m[i,m+1]+D^m[m+1,j])$.

Равенства поясняются достаточно просто. Пусть мы находим кратчайший путь из i в j с промежуточными вершинами из множества $[1..(m+1)]$. Если этот путь не содержит вершину $(m+1)$, то $D^{(m+1)}[i,j]=D^m[i,j]$. Если же он содержит эту вершину, то его можно разделить на две части: от i до $(m+1)$ и от $(m+1)$ до j, а эти оценки сформированы ранее.

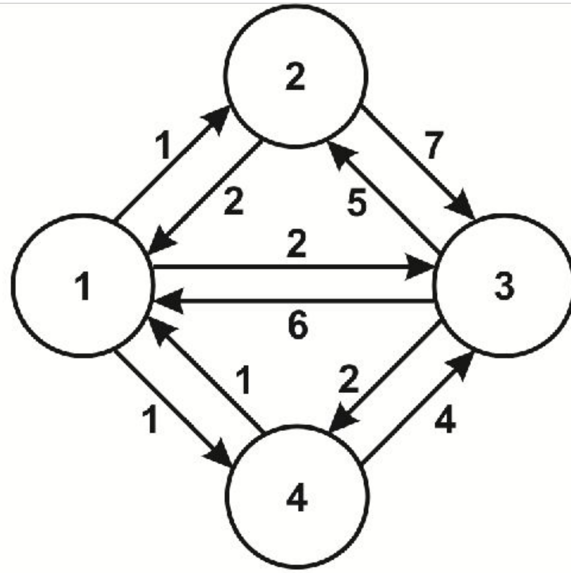

Рис. 12.6. Пример графа для иллюстрации работы алгоритма Флойда

```
Procedure Solve; {A, D - глобальные структуры данных.}
  Var m,i,j:Integer;
  Begin
   For i:=1 To n Do
    For j:=1 To n Do D[i,j]:=A[i,j];
   For i:=1 To n Do D[i,i]:=0;
   For m:=1 To n Do
    For i:=1 To n Do
     For j:=1 To n Do D[i,j]:=Min(D[i,j],D[i,m]+D[m,j]);
  End;
```

Пример. На рис. 12.6 приведен пример графа. Матрица D при работе процедуры *Solve* изменяется следующим образом. Верхний индекс у D указывает номер итерации (значение m).

$$D^0 = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 7 & \infty \\ 6 & 5 & 0 & 2 \\ 1 & \infty & 4 & 0 \end{pmatrix}.\ D^1 = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 4 & 3 \\ 6 & 5 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{pmatrix}.\ D^2 = D^1.\ D^3 = D^2.\ D^4 = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 4 & 3 \\ 3 & 4 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{pmatrix}.$$

Расстояния между парами вершин дает D. Для вывода самих кратчайших путей требуется ввести матрицу M того же типа, что и D. Элемент $M[i,j]$ определяет предпоследнюю вершину кратчайшего пути из i в j. Её изменение в процессе работы *Solve* (требует корректировки) имеет вид:

$$M^0 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{pmatrix}.\ M^1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 3 & 3 & 3 & 3 \\ 4 & 1 & 1 & 4 \end{pmatrix}.\ M^2 = M^1.\ M^3 = M^2.\ M^4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 4 & 1 & 3 & 3 \\ 4 & 1 & 1 & 4 \end{pmatrix}.$$

На первой итерации предпоследней вершиной в путях из 2 в 3, из 2 в 4, из 4 в 2 и из 4 в 3 становится первая вершина. В том случае, когда $D[i,j]$ больше $D[i,m]+D[m,j]$, изменяется не только $D[i,j]$, но и $M[i,j]$. Причем $M[i,j]$ присваивается значение $M[m,j]$, а не m. Поэтому $M[3,2]$ на последней итерации присваивается не 4, а $M[4,2]$. Например, необходимо вывести кратчайший путь из 3-й вершины во 2-ю. Элемент $M[3,2]$ равен 1, поэтому смотрим на элемент $M[3,1]$. Он равен четырем. Сравниваем $M[3,4]$ с 3-й. Есть совпадение, мы получили кратчайший путь: $3\to4\to1\to 2$.

```
Procedure All_Way(i,j:Integer);{Вывод пути между вершинами i и j.}
 Begin
  If M[i,j]=i Then If i=j Write(i) Else Write(i,'-',j)
   Else Begin
          All_Way(i,M[i,j]);{Вывод пути из i в M[i,j].}
          All_Way(M[i,j],j);{Вывод пути из M[i,j] в j.}
         End;
 End;
```