**ЛАБОРАТОРНАЯ РАБОТА № 3.**

***Нахождение кратчайшего пути в графе***

Присвоим каждому ребру графа *вес*, некоторую меру. Например, это может быть расстояние между вершинами (городами в конкретной задаче).

Матрица смежности преобразуется, таким образом, в матрицу весов и элемент  матрицы весов будет равен весу соответствующего ребра (дуги).

Ставится задача: определить кратчайшее расстояние от заданной вершины до всех остальных вершин. Рассмотрим алгоритм Дейкстрыдля решения этой задачи на примере смешанного графа.

Алгоритм позволяет найти кратчайший путь между некоторой вершиной и остальными вершинами при условии, что в графе нет дуг (ребер) с отрицательным весом. Рассмотрим основную идею алгоритма на примере графа на рис. 1.,с матрицей весов дуг рис.2(в которой по строкам и столбцам отмечены номера вершин, а на пересечении *i* строки и *j* столбца указана длинна (вес) дуги (ребра) соединяющей *i-ю* и j*-ю* вершины).Требуется найти все кратчайшие пути от вершины ко всем остальным вершины.

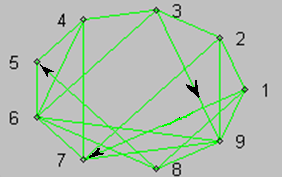


рис.1. рис.2

Суть алгоритма состоит в том, что мы будем искать кратчайший путь сначала от вершины до одной из смежных вершин. Затем от вершин с найденными кратчайшими путями до одной из следующих смежными с ними вершин, пока не найдем кратчайшие пути до всех вершин или до заранее заданной вершины.

Составим вспомогательную таблицу, в которой будем записывать найденные пути от вершины до соответствующих вершин..

*На первом шаге* вычисляем пути от вершины до смежных с ней вершин. Каждый такой путь равен весу дуги (ребра), соединяющейи соответствующую вершину, записываем их в таблицу. Путь от до считаем равным 0 и помечаем его «\*» как далее не вычисляемый. Для вершин,несмежныхс вершиной , путь принимаем равный



В нижнюю строку записываем для найденных путей вершину, от которой высчитывали пути (т.е. ). Впоследний столбец записываем вершину, путь до которой был кратчайшим и величину самого пути (рассматриваются все пути в таблице, за исключением,помеченныхсимволом «\*»). В нашем случае это  и в таблице помечаем ее «\*».

Получаем следующее заполнение таблицы:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  итерации |  |  |  |  |  |  |  |  |  | Кратч. расстояние отдо |
| 1 | 0\* | 10 |  |  |  |  | 3\* | 6 | 12 |  |
| … |  |  |  |  |  |  |  |  |  |  |
| Предшест-вующая вершина |  |  |  |  |  |  |  |  |  |  |

Путь от до ,равный 3, будет кратчайшим в графе, поскольку пути от до всех остальных смежных вершин изначально длиннее.Соответственно, любой обходной путь будет также длиннее.

*На втором шаге* берем вершину для которой на предыдущем шаге был вычислен наикратчайший путь (в нашем случае это ) и вычисляем для смежных с ней вершин (кроме вершин помеченных «\*») пути, равные сумме наикратчайшего пути до и весу дуги ( ребра ) до соответствующей смежной вершины. Далее сравниваем вычисленный путь с путем в таблице. Если вычисленный путь оказался больше, то для данной вершины все оставляем по-прежнему. Если же вычисленный путь оказался меньше, то в таблице записываем новое значение пути для данной вершины, а в последней строке заменяем значение предшествующей вершины на.

После вычисления всех путей к вершинам, смежным с,выбираем кратчайший путь и помечаем соответствующую вершину \*. В последний столбец таблицызаписываем вершину с соответствующим ей кратчайшим из вычисленных путей. В нашем случае это 

Получаем следующее заполнение таблицы:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  итерации |  |  |  |  |  |  |  |  |  | Кратч.  расстояние  от до |
| 1 | 0\* | 10 |  |  |  |  | 3\* | 6 | 12 |  |
| 2 |  | 5\* |  | 7 |  | 17 |  | 6 | 12 |  |
| … |  |  |  |  |  |  |  |  |  |  |
| Предшест-вующая вершина |  |  |  |  |  |  |  |  |  |  |

Далее повторяем действия в шаге 2 для новой вершины, и т.д., до тех пор, пока не найдем кратчайшие пути ко всем вершинам.

В конечном итоге получим следующую таблицу:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  итерации |  |  |  |  |  |  |  |  |  | Кратч.  расстояние  от до |
| 1 | 0\* | 10 |  |  |  |  | 3\* | 6 | 12 |  |
| 2 |  | 5\* |  | 7 |  | 17 |  | 6 | 12 |  |
| 3 |  |  |  | 7 |  | 17 |  | 6\* | 12 |  |
| 4 |  |  | 23 | 7\* | 23 | 17 |  |  | 11 |  |
| 5 |  |  | 23 |  | 12 | 17 |  |  | 11\* |  |
| 6 |  |  | 23 |  | 12\* | 17 |  |  |  |  |
| 7 |  |  | 23 |  |  | 17\* |  |  |  |  |
| 8 |  |  | 23\* |  |  |  |  |  |  |  |
| Предшест-вующая вершина |  |  |  |  |  |  |  |  |  |  |

В данной таблице в крайнем правом столбце перечислены значения кратчайших путей от вершины до всех вершин графа. А для нахождения вершин, через которые проходит кратчайший путь к данной вершине, поступают следующим образом: для вершины к которой необходимо найти путь из последней строки таблицы находится предыдущая вершина, для данной предыдущей вершины из последней строки таблицы находим для нее предыдущую вершину и т.д., до начальной вершины.

Для отрицательных весов используют, например, алгоритм Флойда-Уоршолла

***Задание.***

1. Алгоритмом Дейкстры вычислить кратчайшие пути от вершины  ко всем вершинам графа. Варианты графов указаны в таблице 1. Графы заданы списком ребер, в квадратных скобках указаны веса соответствующих ребер.
2. Алгоритмом Флойда-Уоршолла вычислить кратчайшие пути от вершины ко всем вершинам графа.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| №  № | Кол.  вер-  шин | Кол.  ре-  бер | Ребра  и веса |
| 1. | 6 | 9 | {1,2},{1,4}{2,3},{2,4},{3,4}  {3,5},{3,6},{4,5},{5,6};  [4,3,6,5,4,7,8,5,6]; |
| 2. | 6 | 9 | {1,2],{1,3},{1,4},{2,4},{2,6}  {3,4},{3,5}{4,5}{4,6};  [5,7,4,3,2,5,8,6,4]; |
| 3 | 7 | 10 | {1,2},{1,4},{1,5},{2,3},{2,4},  {3,4},{4,5},{4,6},{6,7},{7,5};  [5,6,8,6,3,6,4,2,3,5]; |
| 4. | 7 | 11 | {1,2},{1,5},{2,3},{2,4},{2,6}.{2,7},  {3,6},{4,5},{4,7},{5,7},{6,7};  [5,3,6,4,7,9,5,3,6,4,2]; |
| 5 | 7 | 11 | {1,2},{1,3},{1,4},{2,4},{3,4},{3,6},  {3,7},{4,5},{4,6},{5,7},{6,7};  [5,4,6,3,3,5,9,5,4,3,6]; |
| 6. | 7 | 11 | {1,2},{1,3},{2,4},{2,5},{2,6}.{2,7},  {2,5},{3,6}{4,6},{5,6},{6,7};  [5,4,3,6,6,8,5,7,4,4,3]; |

***Литература***

1. Андерсон, Д. Дискретная математика и комбинаторика.
2. Липский, В. Комбинаторика для программистов.
3. Кормен Т. Алгоритмы Построение и анализ.