Министерство образования Республики Белару сь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Филиал

«Минский радиотехнический колледж»

Учебная дисциплина «Структуры и алгоритмы обработки данных»

**Инструкция**

по выполнению лабораторной работы

«Реализация алгоритмов поиска кратчайших путей в графе»

Минск

2018

**Лабораторная работа № 10**

**Тема работы: «Реализация алгоритмов поиска кратчайших путей в графе»**

**1. Цель работы**

Формирование умений нахождения кратчайших путей в графе.

**2. Задание**

1. Для заданного графа выполнить поиск кратчайших путей из заданной вершины **s** во все.

**3. Оснащение работы**

ПК, Pascal.

**4. Основные теоретические сведения**

*Пусть дан граф, дугам которого приписаны веса. Задача о нахождении кратчайшего пути состоит в нахождении кратчайшего пути от заданной начальной вершины до заданной конечной вершины, при условии, что такой путь существует.*

***Путем*** *от* ***А1*** *до* ***Аn*** *в графе называется такая последовательность ребер, ведущая от* ***A1*** *к* ***Аn****, в которой каждые два соседних ребра имеют общую вершину и никакое ребро не встречается более одного раза.*

*Вершина* ***А1*** *— начало пути, вершина* ***Аn****— конец.*

Выделяют следующие постановки задачи о кратчайшем пути:

**1. Задача о кратчайшем пути между парой вершин (single-pair shortest path problem).**

Требуется найти кратчайший путь из заданной вершины **s** в заданную вершину **d**.

**2. Задача о кратчайших путях из заданной вершины во все (single-source shortest path problem).**

Найти кратчайшие пути из заданной вершины **s** во все.

*Задача о кратчайших путях с единственным источником.*

**3. Задача о кратчайшем пути в заданный пункт назначения (single-destination shortest path problem).**

Требуется найти кратчайшие пути в заданную вершину v из всех вершин графа.

**4. Задача о кратчайшем пути между всеми парами вершин (all-pair shortest path problem).**

Требуется найти кратчайший путь из каждой вершины **u** в каждую вершину **v**.

**Алгоритм Дейкстры**

Находит кратчайшее расстояние от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса *(*так как на таком цикле бесконечно будет уменьшатся наилучший путь).

На каждом шаге цикла мы ищем вершину с минимальным расстоянием и флагом равным нулю. Затем мы отмечаем её пройденной и проверяем все соседние с ней вершины. Если в ней расстояние больше, чем сумма расстояния до текущей вершины и длины ребра, то уменьшаем его. Цикл завершается когда все вершины будут пройдены.

Алгоритм использует три массива из N (= числу вершин сети) чисел каждый:

1) первый массив **A** содержит метки с двумя значения: **0** (вершина еще не рассмотрена) и **1** (вершина уже рассмотрена);

2) второй массив **B** содержит расстояния – текущие кратчайшие расстояния от **i** до соответствующей вершины. Изначально, B[i]=0, а для всех остальных вершин эта длина равна бесконечности (при реализации на компьютере обычно в качестве бесконечности выбирают просто достаточно большое число, заведомо большее возможной длины пути);

3) третий массив **С** содержит номера вершин - **k**-й элемент **С[k]** есть номер предпоследней вершины на текущем кратчайшем пути из **Vi** в **Vk**.

Алгоритм Дейкстры (Dijkstra)
11
 Пример: найти кратчайший путь
из вершины 1 в вершину 5
 Введем обозначения:
 H – множе...Матрица расстояний **D[i, k]** задает длины дуг **D[i, k]:** если такой дуги нет, то **D[i,k]** присваивается большое число **Б**, равное "машинной бесконечности".

D [i, k]= [0,10,∞,30,100

10,0,50, ∞,∞

∞,50,0,20,10

30, ∞,20,0,60

100, ∞,10,60,0]

**Шаг1:** (инициализация). В цикле от **1** до **N** заполнить нулями массив **A**; заполнить числом **i** массив **C**; перенести i-ю строку матрицы **D** в массив **B**, **A[i]:=1**; **C[i]:=0** (i - номер стартовой вершины)

*i:=1*

*А [0,0,0,0,0] А [0,0,0,0,0]*

*C [i,i,i,i,i] C[1,1,1,1,1]*

*For j=1 to N doB[i]:=d[i,j] B[0,10, ∞,30,100]*

*A[i]:=1 – вершина просмотрена A[1]:=1 // А [1,0,0,0,0]*

*C[i]:=0 – т.к.* ***i*** *– номер стартовой вершины C[i]:=0 // C[0,1,1,1,1]*

**Шаг2:** (общий шаг). Hайти минимум среди неотмеченных (т. е. тех **k**, для которых **A[k]=0**); пусть минимум достигается на индексе **j**, т. е. **B[j] <= B[k].**

*B[0,****10****, ∞,30,100], А [1,0,0,0,0]*

*B[j]=10, j=2*

Затем выполняются следующие операции:

**A[j]:=1;** A[2]:=1;*А [1,1,0,0,0]*

если **B[k] > B[j] + D[j,k]**, то (**B[k] := B[j] + D[j,k]**; **C[k] := j** )

*B[0,****10****,* ***60****,30,100]*

*C[0,1,2,1,1]*

(условие означает, что путь **Vi ... Vk** длиннее, чем путь **Vi...Vj Vk**).

*продолжаем цикл*

*B[0,****10****,* ***60****,30,100] А [1,1,0,0,0]*

*B[j]=30, j=4*

*A[4]:=1* *А [1,1,0,1,0]*

*B[0,****10****,* ***50****,30,100] // B[k]>B[j]+D[j,k], то B[k]:= B[j]+D[j,k]*

*// при k=3 60>30+20, следовательно B[3]:=50*

*C[k]=4, т.к. k=3*

*C[0,1,4,1,1] // C[k] := j*

*продолжаем цикл*

*B[0,10, 50,30,100] А [1,1,0,1,0]*

*B[j]=50, j=3*

*A[3]:=1* *А [1,1,1,1,0]*

*B[0,10, 50,30,****60****] // B[k]>B[j]+D[j,k], то B[k]:= B[j]+D[j,k]*

*// при k=5 100>50+10, следовательно B[5]:=60*

*C[k]=3, т.к. k=5*

*C[0,1,4,1,3]*

*продолжаем цикл*

*B[0,10, 50,30,60] А [1,1,1,1,0]*

*B[j]=60, j=5*

*A[5]:=1* *А [1,1,1,1,1]*

Т.о. сам алгоритм Дейкстры состоит из **N** итераций. На очередной итерации выбирается вершина **j** с наименьшей величиной **B[j]** среди ещё не помеченных.

Выбранная таким образом вершина **j** отмечается помеченной. Далее, на текущей итерации, из вершины **j** производятся релаксации: просматриваются все рёбра D[j,k], исходящие из вершины **j**, и для каждой такой вершины **k** алгоритм пытается улучшить значение **B[k]**.

На этом текущая итерация заканчивается, алгоритм переходит к следующей итерации (снова выбирается вершина с наименьшей величиной **B[k]**, из неё производятся релаксации, и т.д.). При этом в конце концов, после **n** итераций, все вершины графа станут помеченными, и алгоритм свою работу завершает.

Если все **A[k]** отмечены, то длина пути от **Vi** до **Vk** равна **B[k]**. (Теперь надо перечислить вершины, входящие в кратчайший путь).

Стоит заметить, что, если не все вершины графа достижимы из вершины **i**, то значения **B[k]** для них так и останутся бесконечными. Понятно, что несколько последних итераций алгоритма будут как раз выбирать эти вершины, но никакой полезной работы производить эти итерации не будут (поскольку бесконечное расстояние не сможет прорелаксировать другие, даже тоже бесконечные расстояния). Поэтому алгоритм можно сразу останавливать, как только в качестве выбранной вершины берётся вершина с бесконечным расстоянием.

Пример:

**D : array[1..MaxN] of integer; //массив кратчайших расстояний**

**procedure Deisktr(A : Matrix; N, s : integer); //s - искомая вершина**

**var i, j, v, min : longint;**

**begin**

**visited[s] := TRUE; //вершина S посещена**

**for i := 1 to N do D[i] := A[s, i]; //изначальный массив расстояний**

**for i := 1 to n-1 do //на каждом шаге находим минимальное решение и пытаемся его улучшить**

**begin**

**min := inf;**

**for j := 1 to N do if (not visited[j]) and (D[j] < min) then**

**begin**

**min := D[j]; //минимальное расстояние**

**v := j; //найденная вершина**

**end;**

**for j := 1 to N do if (D[j] > D[v] + A[v, j]) and (D[v] < inf) and (A[v, j] < inf) then D[j] := D[v] + A[v, j]; //пытаемся улучшить решение. Если в ней расстояние больше, чем сумма расстояния до текущей вершины и длины ребра, то уменьшаем его.**

**s := v; //новая текущая вершина**

**visited[v] := TRUE; //и она отмечается посещенной**

**end;**

**end;**

**Шаг3:** (выдача ответа). Путь от **Vi** до **Vk** выдается в обратном порядке следующей процедурой:

3.1. z:=C[k];

3.2. Выдать z;

3.3. z:=C[z]. Если z = О, то конец, иначе перейти к 3.2.

Кратчайший маршрут между узлом 1 и любым другим узлом определяется начиная с узла назначения путем прохождения их в обратном направлении с помощью информации, представленной в постоянных метках - **C[k]**. Например, для определения кратчайшего маршрута между узлами 1 и 2 получаем такую обратную последовательность узлов.

*Пример (?):*

*writeln (‘Кратчайший маршрут из вершины’, i)*

*for k:=1 to n do*

*begin*

*writeln (‘в вершину’, k)*

*while z<>0 do*

*begin*

*z:=C[k];*

*write z*

*z:=C[z]*

*end;*

*end*

*Пример:*

**{Dijkstra's algorithm}**

**var a : array [1..20,1..20] of word;**

**c, pred : array [1..20] of word;**

**i, j, k, n, first, last, jn, kn : byte;**

**f, g : text;**

**min : word;**

**Begin**

**assign(f,'in.txt');**

**reset(f);**

**readln(f, n);**

**for i := 1 to n do**

**Begin**

**for j := 1 to n do**

**read(f, a[i,j]);**

**readln(f);**

**end;**

**readln(f, first, last);**

**close(f);**

**min := 32767;**

**for j := 1 to n do**

**if a[first,j] < min then begin min := a[first,j];jn := j;end;**

**c[jn] := min;pred[jn] := first;**

**j := jn;**

**for i := 2 to n do**

**Begin**

**min := 32767;**

**for j := 1 to n do**

**if c[j] <> 0 then**

**Begin**

**for k := 1 to n do**

**if (c[j] + a[j,k] < min)and(c[k] = 0) then begin min := c[j] + a[j,k];jn := j;kn := k;end;**

**end;**

**c[kn] := min;pred[kn] := jn;**

**end;**

**assign(g,'out.txt');**

**rewrite(g);**

**if c[last] = 32767 then writeln(g,'N') else**

**Begin**

**writeln(g,'Y');**

**write(g,first,' ');**

**i := last;k := 1;**

**while i <> first do**

**Begin**

**a[1,k] := i;**

**k := k + 1;**

**i := pred[i];**

**end;**

**for i := k - 1 downto 1 do**

**write(g,a[1,i],' ');**

**writeln(g);**

**writeln(g,c[last]);**

**end;**

**close(g);**

**end.**

Пример:

**unit disktrat;**

**interface**

**const max = 70;**

**Type TJarak = Array [1..max,1..max] of Integer;**

**TPath = record**

**nodeke : byte;**

**Arraypath : Array [1..max] of byte;**

**Jarak : Integer;**

**end;**

**Procedure RuteTerpendek(var Data : TJarak; var Closed : TPath;var awal,tujuan : byte;count : byte);**

**Implementation**

**Procedure RuteTerpendek(var Data : TJarak; var Closed : TPath;var awal,tujuan : byte;count : byte);**

**Var**

**Path : Array [ 1..max] of TPath;**

**Open,Jauh : Array[1..max] of Real ;**

**procedure Initdata(count : byte);**

**var i,j : byte;**

**begin**

**for i := 1 to count do**

**begin**

**open[i] := 0 ;**

**path[i].Jarak := 0;**

**jauh[i] := data[awal,i] ;**

**path[i].arraypath[1] := awal;**

**path[i].nodeke := 1;**

**path[i].arraypath[2] := i;**

**path[i].nodeke := path[i].nodeke + 1;**

**path[i].Jarak :=**

**Data[path[i].arraypath[1],path[i].arraypath[2]]**

**end;**

**open[awal] := awal ;**

**end;**

**procedure update\_close(var Closed: TPath);**

**begin**

**closed := path[Tujuan] ;**

**end ;**

**function successor(count : byte) : byte ;**

**var i,j : byte ;**

**minimum : Real ;**

**begin**

**minimum := 9999 ;**

**for i := 1 to count do**

**if (jauh[i] < minimum) and (open[i] = 0) then**

**begin**

**minimum := jauh[i] ;**

**j := i ;**

**end ;**

**successor := j ;**

**end ;**

**procedure lessthen(count,x : byte) ;**

**var i : byte ;**

**a,b : Real ;**

**begin**

**for i := 1 to count do**

**begin**

**a := jauh[x] + data[x,i] ;**

**b := jauh[i] ;**

**if a < b then**

**begin**

**jauh[i] := a ;**

**path[i] := path[x];**

**path[i].nodeke := path[i].nodeke + 1;**

**path[i].arraypath[path[i].nodeke] := i;**

**path[i].Jarak := path[i].Jarak + Data[path[i].arraypath[path[i].nodeke-1],path[i].arraypath[path[i].nodeke]]**

**end;**

**end ;**

**end ;**

**procedure disktra(count : byte) ;**

**var i,j : byte ;**

**begin**

**for i := 1 to count do**

**begin**

**lessthen(count,successor(count)) ;**

**update\_close(closed);**

**open[successor(count)] := successor(count) ;**

**end ;**

**end ;**

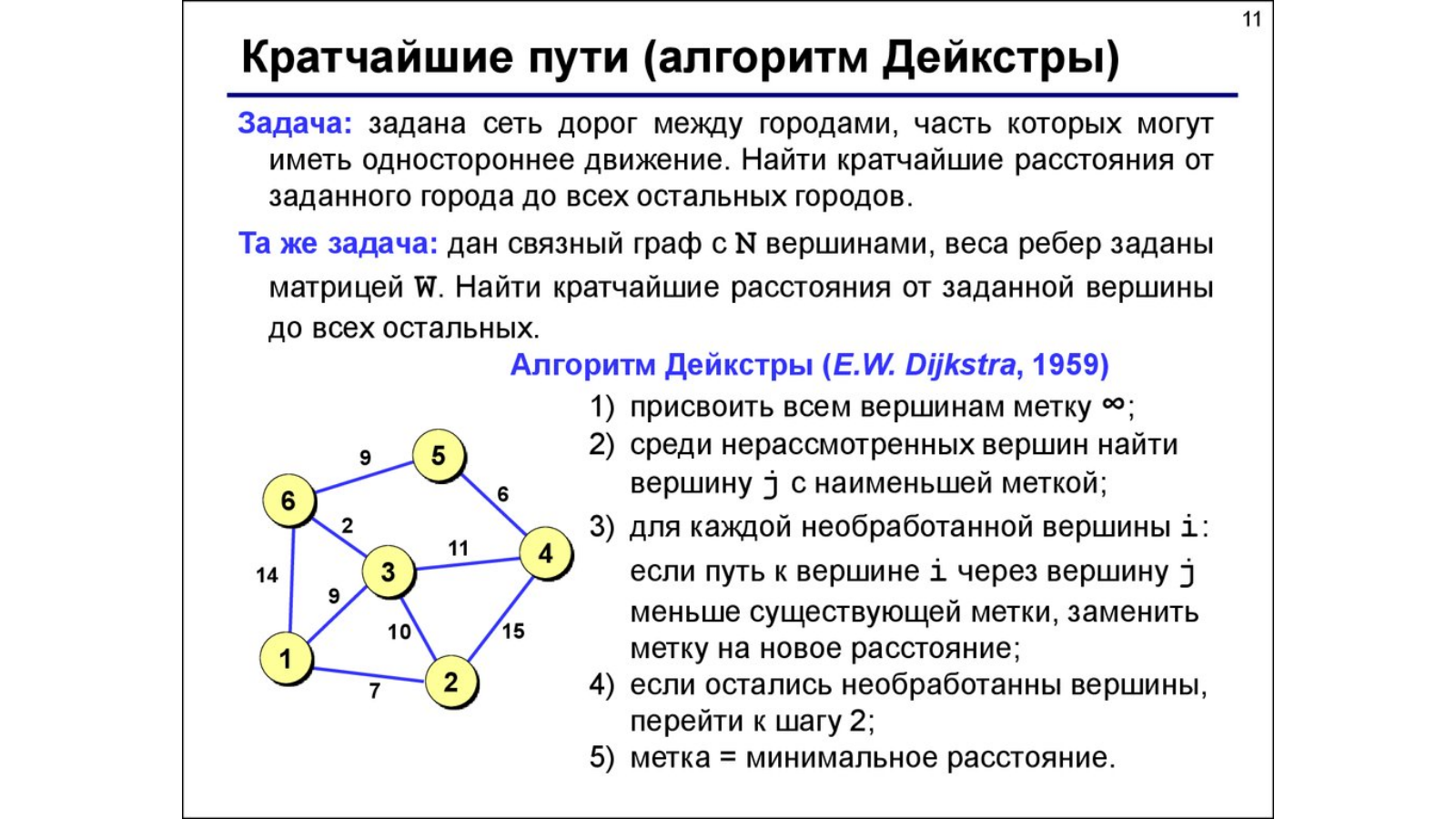
**begin**

**initdata(count);**

**disktra(count);**

**end;**

**end.**



**5. Порядок выполнения работы**

1. Построить граф.

2. Построить для него матрицу смежности

3. Найти кратчайшие пути из заданной вершины вовсе остальные.

4. Вывести маршруты кратчайших путей.

**6. Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7. Контрольные вопросы и задания**

1. Что представляет собой путь в графе?

2. Перечислите виды постановок задач о кратчайшем пути.

3. Опишите основную идею алгоритма Дейкстры.

4. В чем суть алгоритма Флойда?

**8. Рекомендуемая литература**

1.Ахо, А.В.Структуры данных и алгоритмы / А.В. Ахо, Дж.Хопкрофт, Дж. Д.Ульман. – пер. с англ. – М.: Издательский дом "Вильямс", 2007.-400 с.

2. Вирт,Н. Алгоритмы и структуры данных / Н. Вирт. –СПб.:Невский диалект, 2008. – 352с.

3. Вирт, Н. Алгоритмы и структуры данных с примерами на Паскале / Н. Вирт[перевод с английского Д. Б. Подшивалова]; – 2-е изд., испр., – СПб.: Невский Диалект, 2005. – 352с.

4. Гагарина, Л.Г. Алгоритмы и структуры данных / Л.Г. Гагарина, В.Д. Колдаев. – учеб.пособие – М: Финансы и статистика, 2009. – 304с.

5. Котов, В.М. Алгоритмы и структуры данных: учеб.пособие / В.М. Котов, Е.П. Соболевская, А.А. Толстиков – Минск: БГУ, 2011. – 267с.

6. Макконнелл, Дж. Основы современных алгоритмов / Дж. Макконнелл – 2-е дополненное издание – М.:Техносфера, 2006. – 368с.

7. Окулов, С.М. Программирование в алгоритмах / С.М. Окулов. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2007. – 383с.