МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Частное учреждение образования

«Гродненский колледж бизнеса и права»

**Лабораторная работа № 13**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема:** Решение задач на нахождение максимального потока и потока минимальной

для учащихся 2 курса специальности

2-40 01 01 «Программное обеспечение информационных технологий»

**ЛАБОРАТОРНАЯ РАБОТА № 13**

Тема: Решение задач на нахождение максимального потока и потока минимальной.

Цель:

Образовательная**:**

* Обучить основным алгоритмам обхода графа и научиться решать задачи обхода графа на основе поиска в ширину и поиска в глубину,

Развивающая:

* научить анализировать алгоритмы обхода графа и научить решать задачи обхода графа на основе поиска в ширину и поиска в глубину,
* создать условия для развития способности четко формулировать свои мысли.

Воспитательная:

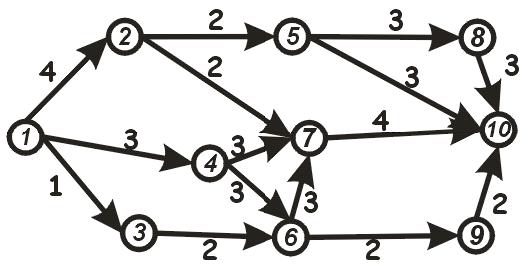
* воспитывать в обучающихся средствами урока уверенность в своих силах;

воспитывать сознательное и серьёзного отношения обучающихся к учебной дисциплине, убеждая их в том, что полученные знания пригодятся им в будущей деятельности.

Задачи: Освоение теоретического материала и выполнение индивидуального задания.

**ЗАДАЧИ**

Условие: На водоочистную станцию поступают сточные воды от 9 разных источников - насосных станций. Определить максимальный объем сточных вод, который может проходить через систему, если сеть и пропускные способности дуг имеют вид:



Алгоритм: Предоставлен преподавателю в письменном виде.

Решение:

**const**

v = 10; n = 2; inf = 100000;

start = 1;

**type**

vektor = **array** [1..V] **of** integer;

**var**

distance: vektor;

count, index, i, l, u, m, z, min: integer;

visited: **array**[1..V]**of** boolean;

f, c: **array**[1..100, 1..100]**of** longint;

ne, cur, next, prev, s, h: **array**[1..100]**of** longint;

**const**

GR: **array** [1..V, 1..V] **of** integer = (

(0, 4, 1, 3, 0, 0, 0, 0, 0, 0),

(0, 0, 0, 0, 2, 0, 2, 0, 0, 0),

(0, 0, 0, 0, 0, 2, 0, 0, 0, 0),

(0, 0, 0, 0, 0, 3, 3, 0, 0, 0),

(0, 0, 0, 0, 0, 0, 0, 3, 0, 3),

(0, 0, 0, 0, 0, 0, 3, 0, 2, 0),

(0, 0, 0, 0, 0, 0, 0, 0, 0, 4),

(0, 0, 0, 0, 0, 0, 0, 0, 0, 3),

(0, 0, 0, 0, 0, 0, 0, 0, 0, 2),

(0, 0, 0, 0, 0, 0, 0, 0, 0, 0));

**procedure** Dijkstra(st: integer);

**var**

count, index, i, u, m, min: integer;

**begin**

m := st;

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

**begin**

distance[i] := inf;

visited[i] := false;

**end**;

distance[st] := 0;

**for** count := 1 **to** V - 1 **do**

**begin**

z := 10 \* n - 5;

u := 0;

min := inf;

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

**if** (**not** visited[i]) **and** (distance[i] <= min) **then**

**begin**

min := distance[i];

index := i;

**end**;

u := index;

visited[u] := true;

l := z - 2;

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

**if** (**not** visited[i]) **and** (GR[u, i] <> 0) **and** (distance[u] <> inf) **and**

(distance[u] + GR[u, i] < distance[i]) **then**

distance[i] := distance[u] + GR[u, i];

**end**;

writeln(' Стоимоть пути из начальной вершины до остальных ');

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

**if** distance[i] <> inf **then**

writeln(m, ' > ', i, ' = ', distance[i])

**else**

writeln(m, ' > ', i, ' = ', ' маршрут недоступен ' );

**end**;

**procedure** MaksEL(u: longint);

**begin**

**while** s[u] > 0 **do**

**begin**

u := distance[cur[l]];

**if** v = 0 **then**

**begin**

cur[l] := 1;

**end**

**else**

**if** (c[l, v] - f[l, v] > 0) **and** (distance[l] = h[v] + 1) **then**

inc(cur[l]);

**end**;

**end**;

**begin**

write(' Начальная вершина >> '); writeln(start);

Dijkstra(start);

writeln(' Максимальный обьем вод - ', l);

**end**.

**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1) При формулировке задачи в пространстве состояний реше¬ние получается в результате применения операторов к описа¬ниям состояний до тех пор, пока не будет получено выражение, описывающее состояние, которое соответствует достижению цели

2) Сущность этих методов станет понятнее, если мы ограничим¬ся рассмотрением деревьев, а не произвольный графов. Дере¬вом называется граф, каждая вершина которого имеет ровно одну непосредственно предшествующую ей (родительскую) вер¬шину, за исключением выделенной вершины, называемой корнем дерева, которая вовсе не имеет предшествующих ей вершин.

3) В методе полного перебора вершины раскрываются в том порядке, в котором они строятся. Простой алгоритм полного перебора на дереве состоит из следующей последовательности шагов:

(1) Поместить начальную вершину в список, называемый ОТКРЫТ.

(2) Если список ОТКРЫТ пуст, то на выход подается сиг¬нал о неудаче поиска, в противном случае переходить к сле¬дующему шагу.

(3) Взять первую вершину из списка ОТКРЫТ и перенести «ее в список ЗАКРЫТ; назовем эту вершину п.

(4) Раскрыть вершину п, образовав все вершины, непосред¬ственно следующие за п. Если непосредственно следующих вер¬шин нет, то переходить сразу же к шагу (2). Поместить имею-щиеся непосредственно следующие за п вершины в конец спи¬ска ОТКРЫТ и построить указатели, ведущие от них назад к вершине n.

(5) Если какие-нибудь из этих непосредственно следующих за п вершин являются целевыми вершинами, то на выход вы¬дать решение, получающееся просмотром вдоль указателей; в противном случае переходить к шагу (2).