МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Лабораторна робота № 6

З дисципліни

“Математичні методи дослідження операцій”

Виконав:

Студент групи КН-314

Ляшеник Остап

Прийняв

Шиманський Володимир Михайлович

Львів - 2023

Постановка завдання

№	4			
0	10	5	14	19
8	0	16	16	8
20	6	0	18	7
9	14	6	0	10
2	13	13	12	0

```
#include <iostream>

#include <algorithm>

#include <vector>

#define INF 100000

#define SIZE 5

using namespace std;

void reduc_matrix(int matrix[SIZE][SIZE], int* d_i, int* d_j, bool* used_i, bool*
used_j){

fill_n(d_i,SIZE,INF);

fill_n(d_j,SIZE,INF);

for(int i = 0; i < SIZE; i++){ //finding d_i vector

for(int j = 0; j < SIZE; j++){

if(matrix[i][j] < d_i[i] && used_i[i] && used_j[j])

d_i[i] = matrix[i][j];

}

}

for(int i = 0; i < SIZE; i++){ //substracting d_i from matrix

for(int j = 0; j < SIZE; j++) {

if(matrix[i][j] != INF && used_i[i] && used_j[j])
```

```
matrix[i][j] -= d_i[i];

}

}

for(int i = 0; i < SIZE; i++){ //finding d_j vector

for(int j = 0; j < SIZE; j++){

if(matrix[i][j] < d_j[j] && used_i[i] && used_j[j])

d_j[j] = matrix[i][j];

}

}

for(int i = 0; i < SIZE; i++){ //substracting d_j from matrix

for(int j = 0; j < SIZE; j++) {

if(matrix[i][j] != INF && used_i[i] && used_j[j])

matrix[i][j] -= d_j[j];

}

}

for(int i = 0; i < SIZE; i++){

if(d_i[i] == INF)

d_i[i] = 0;

if(d_j[i] == INF)

d_j[i] = 0;

}

}

void print_matrix(int matrix[SIZE][SIZE], int* d_i, int* d_j, bool* used_i, bool*
used_j){

cout << "\t";

for(int j = 0; j < SIZE; j++){
```

```
if(used_j[j])
cout << "(" << j+1 << ")\t";
}
cout << "\n";
for(int i = 0; i < SIZE; i++){
if(used_i[i]) {
cout << "(" << i + 1 << ")\t";
for (int j = 0; j < SIZE; j++) {
if(used_j[j]) {
if (matrix[i][j] != INF)
cout << matrix[i][j] << "\t";
else
cout << "inf" << "\t";
}
}
cout << "[" << d_i[i] << "]";
cout << "\n";
}
}
cout << "\t";
for(int j = 0; j < SIZE; j++){
if(used_j[j])
cout << "[" << d_j[j] << "]\t";
}
cout << "\n\n";
```

```
}
int main()
{
int matrix[SIZE][SIZE]{{INF,10,5,14,19}, {8,INF,16,16,8},
{20,6,INF,18,7}, {9,14,6,INF,10}, {2,13,13,12,INF}};
vector<int*> result;
bool used_i[SIZE];
bool used_j[SIZE];
fill_n(used_i,SIZE,1);
fill_n(used_j,SIZE,1);
int d_i[SIZE];
int d_j[SIZE];
reduc_matrix(matrix, d_i, d_j, used_i, used_j);
int bottom_border = 0;
for(int i = 0; i < SIZE; i++)
bottom_border += d_i[i]+d_j[i];
cout << "Start bottom border = " << bottom_border << "\n";
int step = 1;
while(true){
cout << "\n\n---------------------------\nSTEP " << step << "\n\n";
int row, col, max_count(0);
fill_n(d_i,SIZE,INF);
fill_n(d_j,SIZE,INF);
//finding zero with the biggest sum of constants(purple)
for(int i = 0; i < SIZE; i++){
```

```
for(int j = 0; j < SIZE; j++){
if(used_i[i] && used_j[j] && matrix[i][j] == 0){
for (int k = 0; k < SIZE; ++k) { //counting d_i vector
if (matrix[i][k] < d_i[i] && k != j)
d_i[i] = matrix[i][k];
}
for (int k = 0; k < SIZE; ++k) { //counting d_j vector
if (matrix[k][j] < d_j[j] && k != i)
d_j[j] = matrix[k][j];
}
if(max_count < d_i[i] + d_j[j]) {
max_count = d_i[i] + d_j[j];
row = i;
col = j;
}
}
}
}
print_matrix(matrix,d_i, d_j, used_i, used_j);
//excluding node from matrix
matrix[row][col] = INF;
int max_border = bottom_border;
fill_n(d_i,SIZE,INF);
fill_n(d_j,SIZE,INF);
for(int i = 0; i < SIZE; i++){ //finding d_i vector
```

```
for(int j = 0; j < SIZE; j++){
if(matrix[i][j] < d_i[i] && used_i[i] && used_j[j])
d_i[i] = matrix[i][j];
}
}
for(int i = 0; i < SIZE; i++){ //finding d_j vector
for(int j = 0; j < SIZE; j++){
if(matrix[i][j] < d_j[j] && used_i[i] && used_j[j])
d_j[j] = matrix[i][j];
}
}
for(int i = 0; i < SIZE; i++){
if(d_i[i] == INF)
d_i[i] = 0;
if(d_j[i] == INF)
d_j[i] = 0;
}
for(int i = 0; i < SIZE; i++) {
max_border += d_i[i]+d_j[i];
}
cout << "OUR LOW BORDER HERE IS: " << max_border << "\n\n";
matrix[col][row] = INF;
used_i[row] = used_j[col] = 0;
reduc_matrix(matrix, d_i, d_j, used_i, used_j);
print_matrix(matrix,d_i, d_j, used_i, used_j);
```

```
int local_border = bottom_border;
for(int i = 0; i < SIZE; i++) {
local_border += d_i[i]+d_j[i];
}
if(local_border > max_border){
matrix[col][row] = 0;
used_i[row] = used_j[col] = 1;
}else {
result.push_back(new int[2]{row, col}); //adding nodes to result way
cout << "Added: (" << row + 1 << ";" << col + 1 << ")\n\n";
for (int i = 0; i < SIZE; i++)
bottom_border += (d_i[i] + d_j[i]);
cout << "Bottom border = " << bottom_border << " <= " << max_border << "\n";
if (result.size() == SIZE - 2)
break;
step++;
}
}
//adding last two nodes to way
cout << "\nAdding last two nodes: ";
int inf_row, inf_col;
for(int i = 0; i < SIZE; i++){
for(int j = 0; j < SIZE; j++) {
if(matrix[i][j] == INF && used_i[i] && used_j[j]) {
inf_row = i;
```

```
inf_col = j;
}
}
}
for(int i = 0; i < SIZE; i++){
for(int j = 0; j < SIZE; j++) {
if(used_i[i] && used_j[j] && matrix[i][j] != INF && (i == inf_row || j == inf_col)) {
used_i[i] = used_j[j] = 0;
result.push_back(new int[2]{i, j});
cout << "(" << i+1 << ";" << j+1 << ") ";
}
}
}
//printing last result
cout << "\n\n------------------------------\nFinal route: ";
int current_node = result[0][0];
for(int i = 0; i < SIZE; i++){
if(result[i][0] == current_node) {
cout << "(" << result[i][0]+1 << ";" << result[i][1]+1 << ") ";
current_node = result[i][1];
if(result[i][1] == result[0][0])
break;
i = 0;
}
}
```

```
cout << "\nWay cost: " << bottom_border << "\n";
return 0;
}
```

Start bottom border = 35

STEP 1

(1)	(2)	(3)	(4)	(5)		
(1)	inf	5	0	1	14	[1]
(2)	0	inf	8	0	0	[0]
(3)	14	0	inf	4	1	[1]
(4)	3	8	0	inf	4	[3]
(5)	0	11	11	2	inf	[2]
[0]	[5]	[0]	[1]	[1]		

OUR LOW BORDER HERE IS: 41

(1)	(3)	(4)	(5)		
(1)	inf	0	1	14	[0]

(2)	0	inf	0	0	[0]
(4)	3	0	inf	4	[0]
(5)	0	11	2	inf	[0]
[0]	[0]	[0]	[0]		

Added: (3;2)

Bottom border = 35 <= 41

STEP 2

(1)	(3)	(4)	(5)		
(1)	inf	0	1	14	[1]
(2)	0	inf	0	0	[0]
(4)	3	0	inf	4	[3]
(5)	0	11	2	inf	[2]
[0]	[0]	[1]	[1]		

OUR LOW BORDER HERE IS: 38

(1)	(4)	(5)		
(1)	inf	0	13	[1]
(2)	0	0	0	[0]

(5) 0 2 inf [0]

[0] [0] [0]

Added: (4;3)

Bottom border = 36 <= 38

STEP 3

(1) (4) (5)

(1) inf 0 13 [0]

(2) 0 0 0 [0]

(5) 0 2 inf [2]

[0] [0] [1]

OUR LOW BORDER HERE IS: 38

(4) (5)

(1) 0 inf [0]

(2) 0 0 [0]

[0] [0]

Added: (5;1)

Bottom border = 36 <= 38

Adding last two nodes: (1;4) (2;5)

Final route: (3;2) (2;5) (5;1) (1;4) (4;3)

Way cost: 36