In the program:

Height = 200, Width = 250 are constant.

Buffer[y \* width + x] means the weight of cell(x, y).

And dis[x][y] means the length of the shortest path from cell(0, 0) to cell(x, y).

And flag[x][y] means the direction to come cell(x, y) at last in the shortest path.

Dx[] = {0, 1, 0, -1}

Dy[] = {-1, 0, 1, 0}

(dx[0], dy[0]) : N, (dx[1], dy[1]) : E, (dx[2], dy[2]) : S, (dx[3], dy[3]) : W.

Initially, dis[0][0] = weight of cell(0, 0) and the other cells are infinite.

Dis[x][y] = min(dis[x – 1][y], dis[x + 1][y], dis[x][y – 1], dis[x][y + 1]) + weight of cell(x, y).

So we run above algorithm until there is no improvement.

We implement this algorithm by using priority\_queue.

When pq is empty, then dis[x][y] for all cell(x, y) is the length of shortest path to (x, y).

Code :

#include <bits/stdc++.h> or

#include <stdio.h>

#include <algorithm>

#include <queue>

#include <memory.h>

using namespace std;

#define NN 333

#define beyond(x, y) (x<0 || y<0 || x>=width || y>=height)

char raw\_name[100]="map1.raw";

int height=200,width=250;

unsigned char buffer[NN\*NN];

// read map with size(W\*H)

void inline read(const char\* filename, int W, int H){

FILE\* fp=fopen(filename,"rb");

fread(buffer, W, H , fp);

}

// input the map file name and the size of map

void input(){

scanf("%s %d %d",raw\_name,&height,&width);

}

int dx[]={0, 1, 0, -1};

int dy[]={-1, 0, 1, 0};

int dis[NN][NN];

int flag[NN][NN];

typedef pair <int, int> pii;

priority\_queue <pii, vector <pii>, greater <pii> > pq;

// calulate the shortest path from cell(0, 0) to all others

void solve(){

memset(dis, 0x3f, sizeof(dis));

dis[0][0]=buffer[0];

pq.push(pii(dis[0][0], 0));

while(!pq.empty()){

pii tmp=pq.top();

pq.pop();

int y=tmp.second/width;

int x=tmp.second%width;

// consider four cells which are reachable from the cuurent cell(x, y)

for(int i=0; i<4; i++){

int xx=x+dx[i];

int yy=y+dy[i];

// determine that the cell(xx, yy) is in the map

if(beyond(xx, yy)) continue;

if(dis[yy][xx]>dis[y][x]+buffer[yy\*width+xx]){

dis[yy][xx]=dis[y][x]+buffer[yy\*width+xx];

pq.push(pii(dis[yy][xx], yy\*width+xx));

flag[yy][xx]=i+1;

}

}

}

}

int main(){

//input();

read(raw\_name,width,height);

/\*

freopen("out.txt","w",stdout);

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

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

printf("%d ", buffer[i\*width+j]);

}

puts("");

}

fclose(stdout);

\*/

solve();

freopen("ShortestPath.txt","w",stdout);

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

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

printf("%d ", dis[i][j]); //50,100,150,200

}

puts("");

}

fclose(stdout);

freopen("path.raw","wb",stdout);

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

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

printf("%c", 50\*flag[i][j]); //50,100,150,200

}

}

fclose(stdout);

return 0;

}

Input Data : map1.raw

Output Data :

Distance : Shortest path from a point to all others.pptx

Direction : path.raw