**ALGORITHM**

**201133216**

**정유석**

Problem 5 : The Cheapest Way

**CODE**

#include<stdio.h>

#include<stdlib.h>

#define MAX\_ROW 10 //행

#define MAX\_COLUMN 100//열

typedef struct { //table of structure has parent table and mark table

int parent;

int mark;

}table;

table t[MAX\_COLUMN][MAX\_ROW]; //Make a big table

void main(){

char string[100]; //Receive the row components

int cmp\_R, cmp\_C ,cmp\_MIN; //Compare value

int i,j, M\_C, M\_R; // i,j loop value, and M\_C, M\_R are Max column, and Max Row.

int number[100], cnt=0, min, min\_i, min\_j; // number of matrix is store the routine of mark table , cnt is count the number

int find\_C, find\_R, find\_Min; // min, min\_i, min\_j, find\_C, find\_R, find\_Min are used to find the routine of mark table (compare the parent table)

printf("Input Column and Row : "); //Input

scanf("%d %d",&M\_C,&M\_R);

while(M\_C>MAX\_COLUMN || M\_C<1 || M\_R>MAX\_ROW || M\_R<1){ //Exception

printf("Wrong value ! Re input (0<Column<100, 0<Row<10) : ");

scanf("%d %d",&M\_C,&M\_R);

}

fflush(stdin);

M\_C--; //So (5,6) --> column = 4, row = 5 in the program

M\_R--;

for(i=0; i<=M\_C; i++){

printf("Input the Mark value : "); //input mark component of row

gets(string);

fflush(stdin);

for(j=0; j<=M\_R; j++){

if(j==0) //store the integer from character

t[i][j].mark = string[j]-48;

else

t[i][j].mark = string[j\*2]-48;

}

}

printf("MARK table \n");

for(i=0; i<=M\_C; i++){

for(j=0; j<=M\_R; j++){ //Mark table

printf("%d ",t[i][j].mark);

}

printf("\n");

}

for(i=0; i<=M\_R; i++){

for(j=0; j<=M\_C; j++){

cmp\_MIN = 0;

if(i==0){ //Store the first column - (0,0) (1,0) (2,0) ... (n,0)

t[j][i].parent = t[j][i].mark;

continue;

}

else if(j == 0 && i > 0){ //compare the other componet - if column is n, row is m

cmp\_MIN = t[j][i-1].parent;

if(cmp\_MIN > t[j+1][i-1].parent)

cmp\_MIN = t[j+1][i-1].parent;

if (cmp\_MIN > t[M\_C][i-1].parent)

cmp\_MIN = t[M\_C][i-1].parent;

t[j][i].parent = t[j][i].mark + cmp\_MIN;

}

else if(j <M\_C && j > 0 && i > 0 ){

cmp\_MIN = t[j][i-1].parent;

if(cmp\_MIN > t[j-1][i-1].parent)

cmp\_MIN = t[j-1][i-1].parent;

if(cmp\_MIN > t[j+1][i-1].parent)

cmp\_MIN = t[j+1][i-1].parent;

t[j][i].parent = t[j][i].mark + cmp\_MIN;

}

else if(j == M\_C){

cmp\_MIN = t[j][i-1].parent;

if(cmp\_MIN > t[j-1][i-1].parent)

cmp\_MIN = t[j-1][i-1].parent;

if(cmp\_MIN > t[0][i-1].parent)

cmp\_MIN = t[0][i-1].parent;

t[j][i].parent = t[j][i].mark + cmp\_MIN;

}

else

t[j][i].parent = 0;

}

}

//Display parent table

printf("\n\n\n-----------Parent\n");

for(i=0 ; i<=M\_C; i++){

for(j=0; j<=M\_R; j++){

printf("%d ",t[i][j].parent);

}

printf("\n");

}

printf("-------------------------\n\n");

//Find the minimum cost of way ( last column of parent table)

find\_R=M\_R;

find\_C=0;

find\_Min = t[find\_C][find\_R].parent;

for(i=0; i<=M\_C; i++){

if(find\_Min > t[i][find\_R].parent){

find\_Min = t[i][find\_R].parent;

find\_C = i;

}

}

//find the Routine of parent table

number[cnt] = t[find\_C][find\_R].mark;

while(1){

find\_Min = t[find\_C][find\_R-1].parent;

min\_i = find\_C;

min\_j = find\_R-1;

if(find\_C == 0){ //compare the table

if(min > t[find\_C+1][find\_R-1].parent){

min = t[find\_C+1][find\_R-1].parent;

min\_i = find\_C+1;

min\_j = find\_R-1;

}

if(min > t[M\_C][find\_R-1].parent){

min = t[M\_C][find\_R-1].parent;

min\_i = M\_C;

min\_j = find\_R-1;

}

}

else if(find\_C < M\_C && find\_C >0){

if(find\_Min > t[find\_C-1][find\_R-1].parent){

min = t[find\_C-1][find\_R-1].parent;

min\_i = find\_C-1;

min\_j = find\_R-1;

}

if(min > t[find\_C+1][find\_R-1].parent){

min = t[find\_C+1][find\_R-1].parent;

min\_i = find\_C+1;

min\_j = find\_R-1;

}

}

else if(find\_C == M\_C){

if(find\_Min > t[find\_C-1][find\_R-1].parent){

min = t[find\_C-1][find\_R-1].parent;

min\_i = find\_C-1;

min\_j = find\_R-1;

}

if(min > t[0][find\_R-1].parent){

min = t[0][find\_R-1].parent;

min\_i = 0;

min\_j = find\_R-1;

}

}

find\_C = min\_i;

find\_R = min\_j;

number[++cnt] = t[find\_C][find\_R].mark;

if(find\_C<0 || find\_R<0)

break;

}

//Display routine

for(i= cnt-1; i>=0; i--)

printf("%d ", number[i]);

printf("\n The Cheapest Way : %d\n\n",t[M\_C][M\_R]);

}

**PRINT**

