滑雪问题

10153903105

杜云滔

1.使用DFS和DP方法完成

2.Main.cpp

#include <iostream>

#include "skiing.h"

using namespace std;

int main()

{

int max\_dis=0,temp;

skining new\_skiing;

for (int i=1;i<=new\_skiing.Row;i++)

{

for (int j=1;j<=new\_skiing.Column;j++)

{

new\_skiing.calculate(i,j,0);

temp=new\_skiing.dis(i,j);

max\_dis=max\_dis>temp?max\_dis:temp;

}

}

cout<<"the farthest length is "<<new\_skiing.len<<endl;

cout<<"the farthest length is "<<max\_dis<<endl;

return 0;

}

3.skiing.cpp

#include <iostream>

#include "skiing.h"

using namespace std;

skining::skining()

{

cout<<"please enter the R/C"<<endl;

cin>>Row>>Column;

for (int i=1; i<=Row; i++)

{

for (int j=1; j<=Column; j++)

{

cin>>field[i][j];

dis\_sk[i][j]=0;

}

}

cout<<"finish init!"<<endl;

len=0; //初始化高度

}

int skining::test\_bound(int R,int C)

{

int flag=0;

int new\_R,new\_C;

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

{

//得出下一步的点

new\_R=R+dir[0][i];

new\_C=C+dir[1][i];

if (new\_C>=1 && new\_C<=Column &&new\_R>=1 && new\_R<=Row)

{

//如果下一步的点不是边界

if (field[new\_R][new\_C]!=10001 && field[new\_R][new\_C]<=field[R][C])

{

//如果下一步的点没有走过并且高度更低

flag=1;

break;

}

}

}

return !flag;

}

void skining::calculate(int R,int C,int milage)

{

int temp\_high;

int new\_R,new\_C;

if (test\_bound(R,C))

{

len=milage>len?milage:len;

return;

}

else

{

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

{

//得出下一步的点

new\_R=R+dir[0][i];

new\_C=C+dir[1][i];

if (new\_C>=1 && new\_C<=Column &&new\_R>=1 && new\_R<=Row)

{

//如果下一步的点不是边界

if (field[new\_R][new\_C]!=10001 && field[new\_R][new\_C]<field[R][C])

{

milage+=1;

temp\_high=field[R][C];

field[R][C]=10001;

// cout<<"next step "<<new\_R<<" \* "<<new\_C<<endl;

// cout<<"step =="<<milage<<endl;

calculate(new\_R,new\_C,milage);

field[R][C]=temp\_high;

milage-=1;

}

}

}

}

}

int skining::dis(int C,int R)

{

int temp;

int new\_R,new\_C;

if (dis\_sk[C][R]) return dis\_sk[C][R];//如果已经存在，则直接返回

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

{

new\_C=C+dir[0][i];

new\_R=R+dir[1][i];

if (in\_bound(new\_C,new\_R))//如果在界内

{

if (field[C][R]>field[new\_C][new\_R])//如果可以顺着滑下来

{

temp=dis(new\_C,new\_R);

dis\_sk[C][R]=dis\_sk[C][R]>temp?dis\_sk[C][R]:temp+1;

}

}

}

return dis\_sk[C][R];

}

int skining::in\_bound(int R,int C)

{

return R>=1 && R<=Row && C>=1 && C<=Column;

}

4.skiing.h

#ifndef SKIING\_H\_INCLUDED

#define SKIING\_H\_INCLUDED

static int dir[2][4]={1,0,-1,0,0,1,0,-1};

class skining

{

public:

int field[50][50];

int Row;

int Column;

int len;

int dis\_sk[50][50];

skining();

void calculate(int R,int C,int milage);

int test\_bound(int R,int C);

/////

int in\_bound(int R,int C);

int dis(int i,int j);

/////

};

#endif // SKIING\_H\_INCLUDED