LAB 6

1. INTRODUCTION

This lab requires to write a C program to help Professor Patt, that is, to read a map and tells the longest distance. The map is a N X M matrix, and Patt can only ski to the adjacent vertex only when the height of the adjacent vertex is lower.

To achieve the goal, the program should include a recursive structure. When Patt reaches a vertex, the program should grope the adjacent four vertices to see if there is a path and save the length. The program should repeat this process until every vertex has been set as the beginning once.

2. ALGORITHM

To finish the tasks, the algorithm should be like:

1. Do the initialization;

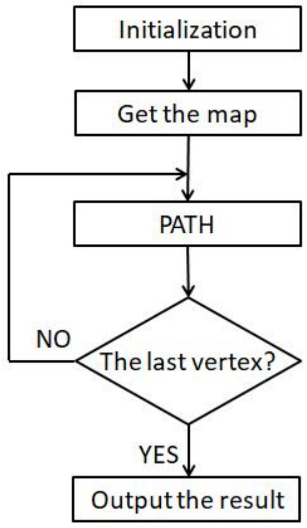
2. Execute the main codes and start the recursion;

3. Test the longest path started from a certain vertex and renew the max length;

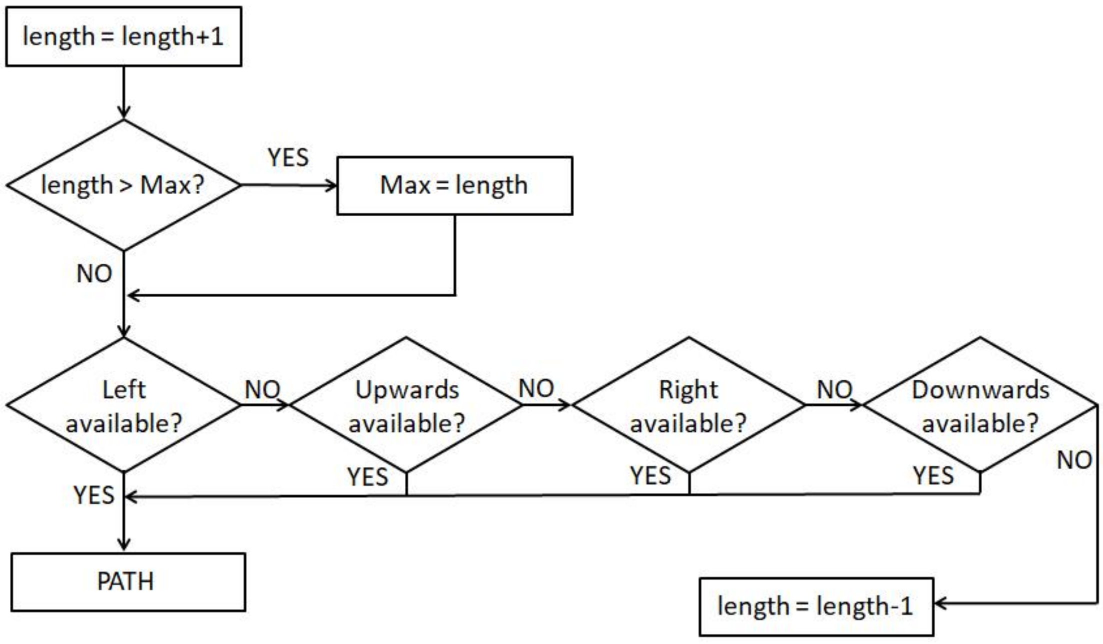
4. Change the starting point and loop until every vertex has been tested;

5. Output the max length;

The diagram of the main codes is shown as follow:

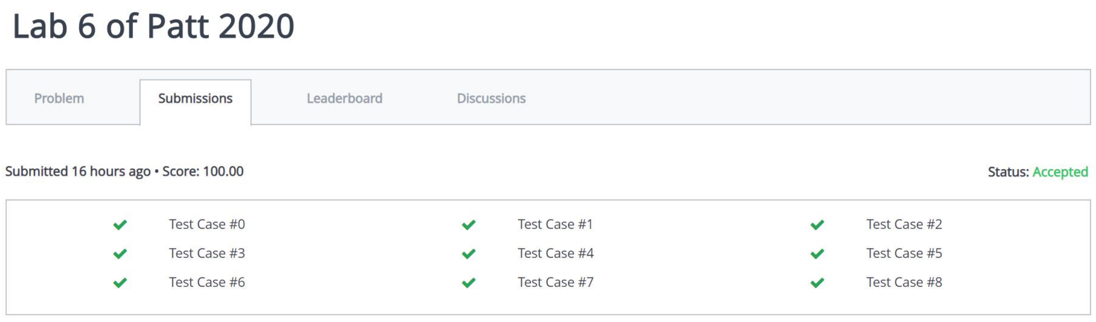


The diagram of the PATH function is shown as follow:



3. Testing Result

The program passes all the test cases on HackerRank.


Lab 6 of Patt 2020

Problem | Submissions | Leaderboard | Discussions

Submitted 16 hours ago • Score: 100.00 Status: Accepted

✓ Test Case #0	✓ Test Case #1	✓ Test Case #2
✓ Test Case #3	✓ Test Case #4	✓ Test Case #5
✓ Test Case #6	✓ Test Case #7	✓ Test Case #8

4. Discussion and Experience

When writing the program, I found that appropriate functions could make the program dramatically more readable.

I deeply agree that a high-level language is much more preferable than an assembly language.

Appendix: Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int map[52], row, colm, Max, length;
void Path(int i);
int Cleft(int i);
int Cup(int i);
int Cright(int i);
int Cdown(int i);
int main()
{
   Max=0;
   length=0;
   scanf("%d %d", &row, &colm);
   for(int i=0; i<row*colm; i++)     /*get the input*/
   {
      scanf("%d", &map[i]);
   }
   for(int i=0; i<row*colm; i++)
   {
      Path(i);
   }
   printf("%d", Max);
   return 0;
```

```
}

void Path(int i)
{
   length++;
   if (length>Max)
   {
      Max=length;      /*renew the max length*/
   }
   if (Cleft(i))
   {
      Path(i-1);
   }
   if (Cup(i))
   {
      Path(i-colm);
   }
   if (Cright(i))
   {
      Path(i+1);
   }
   if (Cdown(i))
   {
      Path(i+colm);
   }
   length--;
}

int Cleft(int i)      /*check if Patt can ski left*/
{
   if ((i%colm) && (map[i-1]<map[i]))
   {
      return 1;
   }
   return 0;
}

int Cup(int i)        /*check if Patt can ski upwards*/
{
   if((i>=colm) && (map[i-colm]<map[i]))
   {
      return 1;
   }
   return 0;
```

```
}

int Cright(int i)     /*check if Patt can ski right*/
{
   if (((i+1)%colm) && (map[i+1]<map[i]))
   {
      return 1;
   }
   return 0;
}

int Cdown(int i)      /*check if Patt can ski downwards*/
{
   if ((i<(row-1)*colm) && (map[i+colm]<map[i]))
   {
      return 1;
   }
   return 0;
}
```