Lab 5: Skiing

1 Algorithm

```
// define the map
int map[M][N] = {
     {89, 88, 86, 83},
     {79, 73, 90, 80},
     {60, 69, 73, 77}
};

// XY coordinate starting position
int startY = 3;
int startX = 4;

// recursive function function to find the longest path
int findLongestPath(int y, int x) {
     // End condition: if the height of the current position is lower than the
surrounding points, return 1
     if (y == 0 || y == M - 1 || x == 0 || x == N - 1 || map[y][x] >= map[y-1]
[x] && map [y][x] >= map[y+1][x] && map[y][x] >= map[y][x-1] && map[y][x] >=
map[y] [x+1]) {
         return 1;
     }

     // four directions and find the longest path among them
     int up = 0, down = 0, left = 0, right = 0;
     if (map[y][x] > map[y-1][x]) {
         up = find_Longest_Path(y-1, x);
     }
     if (map[y][x] > map[y+1][x]) {
         down = find_Longest_Path(y+1, x);
     }
     if (map[y][x] > map[y][x-1]) {
         left = find_Longest_Path(y, x-1);
     }
     if (map[y][x] > map[y][x+1]) {
         right = find_Longest_Path(y, x+1);
     }

     // Return the longest path in the four directions +1
     return 1 + max(up, max(down, max(left, right)));
}

// main program
int main() {
     // Let the function find the longest path and output the result
     int longestPath = findLongestPath(startY, startX);
     // output the longest path
     print(longestPath);
     return 0;
}
```

1. **Define the graph data** : First, we need to define a two-dimensional array representing the graph data. The size of this array is X rows and Y columns, and each element represents the height or altitude of the graph.
2. **Find a starting location** : Select a starting location as the starting point of the search. In this problem, we need to find the lowest point as the starting position, because we want to find the longest path from the lowest point.
3. **Define a recursive function** : We need to define a recursive function to find the longest path from the current location. This function will search up, down, left, and right from the current position in four directions, and record the path length in each direction.
4. **Set End Condition** : In a recursive function, we need to set an end condition. When the border of the graph is reached or the height of the current position is not lower than the surrounding position, we consider that the end of the path has been reached, and 1 should be returned at this time.
5. **Recursive call** : In the recursive function, we will recursively call itself, search from the four directions of up, down, left, and right, and find the longest path length among them.
6. **Return result** : Finally, we return the length of the longest path in the four directions plus 1 (plus 1 because the current location is also counted as a path) as the longest path length starting from the current location.
7. **Main program** : In the main program, we call the recursive function and pass in the coordinates of the starting position. Then, output the returned longest path length as the final result.

The general idea is to start from the lowest point in a recursive manner, search four directions, find the longest path in each direction, then compare the longest paths in these four directions, and finally get the longest path length of the entire graph. This algorithm is guaranteed to cover all possible paths in the graph and find the longest one.

2 Esential Part

- **Traversing the four directions of up, down, left, and right at the current position**

```
UP                  ; 向上
    ADD R4, R4, #0  ; 判断是否在最顶上
    BRnz DOWN
    LDR R1, R0, #0  ; X存到R1
    ADD R4, R4, #-1
    LD R7, xx
    ADD R0,R0,R7
    ADD R7,R0,#0
    LDR R7,R7,#0    ; R0-x存到R7
    NOT R7,R7
    ADD R7,R7,#1
    ADD R7,R1,R7
    BRnz UP_NOW     ; 判断向上的海拔和当前海拔大小

    ADD R3,R3,#1
    JSR FIND_ROAD   ; 返回递归
    ADD R3,R3,#-1
UP_NOW
    LDI R7,X
    ADD R0,R7,R0
    ADD R4,R4,#1    ; 保存R0 R1
```

The other three directions are the same as this code.

3 Questions

Q : Can you roughly describe your recursive algorithm ?

A : Use one register to store the current location address, two to store the coordinate address, the current location is used to get the value, and the coordinates are used to judge whether it has reached the boundary (this is easier), use a large loop to traverse all the points on the map, and then set a Recursion, find the route in the order of up, down, left, and right, take a step and add 1 to the end, then go back and finally find the largest R2.

Q : Faced with an N*M graph, what is its maximum recursion depth ?

A : The initial value is 1, add one to the four directions, and add one if the four directions are satisfied, so the depth of an M*N map is M*N.