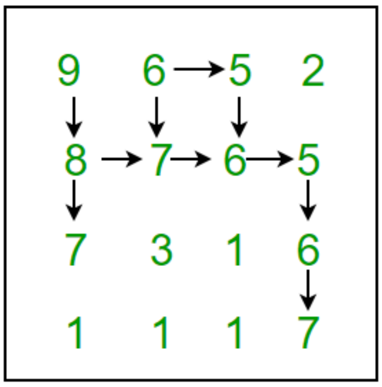
Find maximum length Snake sequence

> 输出最长的蛇形序列（下个数必须是上个数字的+1或者-1）

子问题：最长的蛇形序列长度

dp[i][j] 通过dp[i - 1][j] 或者 dp[i][j - 1]转移得到



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

using namespace std;

int maxSnakeSeq(vector<vector<int>> &matrix, int &N, int &M) {
    vector<vector<int>> dp(N, vector<int>(M));
    int maxLen = INT_MIN;
    dp[0][0] = 1;
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < M; ++j) {
            if (i || j) {
                if (i > 0 && abs(matrix[i - 1][j] - matrix[i][j]) == 1) {
                    dp[i][j] = max(dp[i - 1][j] + 1, dp[i][j]);
                }
                if (j > 0 && abs(matrix[i][j - 1] - matrix[i][j]) == 1) {
                    dp[i][j] = max(dp[i][j - 1] + 1, dp[i][j]);
                }
            }
            maxLen = max(maxLen, dp[i][j]);
        }
    }
    return maxLen;
}
int main() {
    vector<vector<int>> matrix = {{9, 6, 5, 2},
                                  {8, 7, 6, 5},
                                  {7, 3, 1, 6},
                                  {1, 1, 1, 7}};
    int N = 4, M = 4;
    auto maxLen = maxSnakeSeq(matrix, N, M);
    printf("%d\n", maxLen);
    return 0;
}
```

输出蛇形序列

在计算出dp数组以后，记录下最大的dp值的i，j坐标

然后从后往前将matrix[i][j]加入数组，期间需要判断（i，j）的下一个是（i - 1，j）还是（i，j - 1），最后逆置整个数组

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

using namespace std;

vector<int> maxSnakeSeq(vector<vector<int>> &matrix, int &N, int &M) {
    vector<vector<int>> dp(N, vector<int>(M));
    vector<int> path;
    int maxLen = INT_MIN, maxRowIdx = 0, maxColIdx = 0;
    dp[0][0] = 1;
            // 确定最长的蛇形序列长度以及结尾坐标
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < M; ++j) {
            if (i || j) {
                if (i > 0 && abs(matrix[i - 1][j] - matrix[i][j]) == 1) {
                    dp[i][j] = max(dp[i - 1][j] + 1, dp[i][j]);
                }
                if (j > 0 && abs(matrix[i][j - 1] - matrix[i][j]) == 1) {
                    dp[i][j] = max(dp[i][j - 1] + 1, dp[i][j]);
                }
            }
            if (maxLen < dp[i][j]) {
                maxLen = dp[i][j];
                maxRowIdx = i;
                maxColIdx = j;
            }
        }
    }
            // 找到蛇形序列
    path.push_back(matrix[maxRowIdx][maxColIdx]);
    while (maxRowIdx || maxColIdx) {
        if (maxRowIdx > 0 && dp[maxRowIdx][maxColIdx] - 1 == dp[maxRowIdx - 1][maxColIdx]) {
            path.push_back(matrix[maxRowIdx - 1][maxColIdx]);
            maxRowIdx--;
        } else if (maxColIdx > 0 && dp[maxRowIdx][maxColIdx] - 1 == dp[maxRowIdx][maxColIdx - 1]) {
            path.push_back(matrix[maxRowIdx][maxColIdx - 1]);
            maxColIdx--;
        }
    }
    reverse(path.begin(), path.end());
    return path;
}

int main() {
    vector<vector<int>> matrix = {{9, 6, 5, 2},
                                  {8, 7, 6, 5},
                                  {7, 3, 1, 6},
                                  {1, 1, 1, 7}};
    int N = 4, M = 4;
    auto path = maxSnakeSeq(matrix, N, M);
    for (auto p: path)
        printf("%d ", p);
    return 0;
}
```