# Counting Valleys ☆





Editorial by pkacprzak

Our goal is to count the number of valleys. A valley is a sequence of steps starting with a step downward from sea level and ending with a step upward to sea level. Let $level$ be a variable denoting the current altitude. If we take a step upwards, $level$ is incremented by one; if we take step downwards, $level$ is decremented by one.

Since we know that the sequence of input steps starts and ends at sea level, then we can say that our $level$ variable is $0$ at the beginning and end of the hike. The number of valleys can be counted as the number of steps taken upwards to sea level (i.e., when $level$ goes from $-1$ to $0$. This is true, because each such step ends the sequence of steps below sea level, signifying the end of a valley.

Set by pkacprzak

Problem Setter's code:

**C++**

```
#include <iostream>
#include <cstdio>
#include <string>
#include <sstream>
#include <vector>
#include <set>
#include <map>
#include <queue>
#include <stack>
#include <cmath>
#include <algorithm>
#include <cstring>
#include <ctime>
#include <cassert>
using namespace std;
#define pb push_back
#define mp make_pair
#define pii pair<int,int>
#define vi vector<int>
#define vpii vector<pii>
#define SZ(x) ((int)(x.size()))
#define fi first
#define se second
#define FOR(i,n) for(int (i)=0;(i)<(n);++(i))
#define FORI(i,n) for(int (i)=1;(i)<=(n);++(i))
#define IN(x,y) ((y).find((x))!=(y).end())
#define ALL(t) t.begin(),t.end()
#define FOREACH(i,t) for (typeof(t.begin()) i=t.begin(); i!=t.end(); i++)
#define REP(i,a,b) for(int (i)=(a);(i)<=(b);++i)
#define REPD(i,a,b) for(int (i)=(a); (i)>=(b);--i)
#define REMAX(a,b) (a)=max((a),(b));
#define REMIN(a,b) (a)=min((a),(b));
#define DBG cerr << "debug here" << endl;
#define DBGV(vari) cerr << #vari<< " = "<< (vari) <<endl;

typedef long long ll;

const int MINN = 2;
const int MAXN = 1e6;

int main()
{
```

```
    ios_base::sync_with_stdio(0);
    int n;
    cin >> n;
    assert(n >= MINN && n <= MAXN);
    string s;
    cin >> s;
    assert(s.length() == n);
    int res = 0;
    int level = 0;
    FOR(i, n)
    {
        if(s[i] == 'D')
        {
            --level;
        }
        else if(s[i] == 'U')
        {
            ++level;
            if(level == 0) ++res;
        }
    }
    assert(level == 0);
    cout << res << endl;
}
```

 Tested by shashank21j

Problem Tester's code:

### Python 2

```
# Solution uses a two pointer technique,
# we check if current step leads to height 0 and previous height was -ve.
height = 0
prev_height = 0
cnt = 0
n = input()
s = raw_input().strip()
for i in range(len(s)):
    if (s[i] == 'U'):
        height += 1
    elif s[i] == 'D':
        height -= 1
    if height == 0 and prev_height < 0:
        cnt += 1
    prev_height = height
print cnt
```

### Java

```
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        char[] hike = scan.next().toCharArray();

        int count = 0;
        int altitude = 0;

        for(char c : hike) {
            // Step up
            if(c == 'U') {
                if(altitude == -1) {
                    count++;

                }
```

```
                altitude++;
            }
            // Step down
            else {
                altitude--;
            }
        }

        scan.close();

        System.out.println(count);
    }
}
```

### Swift

```
var n = Int(readLine()!)!
var hike = readLine()!
var numValleys = 0
var altitude = 0

for char in hike.characters {
    if(char == "U") {
        if(altitude == -1) {
            numValleys += 1;
        }
        altitude += 1;
    }
    // Step down
    else {
        altitude -= 1;
    }
}

print(numValleys)
```

### Feedback

Was this editorial helpful?

Yes No