Sherlock and Cost

LK Editorial by Lalit Kundu

Problem: Array A contains the elements $A_1, A_2 \dots A_N$. And array B contains the elements $B_1, B_2 \dots B_N$. There is a relationship between A_i and $B_i \forall 1 \leq i \leq N$. That is, for any valid i, $1 \leq A_i \leq B_i$.

The cost S of A is defined as:

$$S = \sum_{i=2}^{N} |A_i - A_{i-1}|$$

Print the largest possible value of S.

Solution: For S to be largest, A_i will either be 1 or B_i because $|1 - x| > (|2 - x| = |1 - (x - 1)|)$. We can make a simple DP now, say $DP[n][2]$. $DP[i][0]$ stores the maximum value of S using the first i elements only if A_i was 1. If $A[i] = B[i]$, $DP[i][1]$ stores the maximum value of S possible using the first i elements only.

See setter's solution for more details.

LK Set by Lalit Kundu

Problem Setter's code:

C++

```
#include<bits/stdc++.h>
using namespace std;
int ar[100005]={},dp[100005][2]={};
int main()
{
    int t;
    cin >> t;
    while(t--)
    {
        int n,i,j;
        cin >> n;
        for(i=0; i<n; i++)
            cin >> ar[i];
        for(i=0; i<n-1; i++)
        {
            dp[i+1][0]=max(dp[i][0],dp[i][1]+abs(ar[i]-1));
            dp[i+1][1]=max(dp[i][0]+abs(ar[i+1]-1),dp[i][1]+abs(ar[i]-ar[i+1]));
        }
        cout << max(dp[n-1][0],dp[n-1][1]) << endl;
    }
    return 0;
}
```

Tested by gera1d

Problem Tester's code:

C++

```
#ifdef ssu1
#define _GLIBCXX_DEBUG
#endif
#undef NDEBUG

#include <algorithm>
#include <functional>
#include <numeric>
#include <iostream>
#include <cstdio>
#include <cmath>
```

STATISTICS

Difficulty:	Medium
Success Rate:	52.02%
Time Complexity:	O(N)
Required Knowledge:	Dynamic Programming
Publish Date:	May 30 2014

Originally featured in 101 Hack May

Of the 1164 contest participants, 198 (17.01%) submitted code for this challenge.

NEED HELP?

View discussions

View top submissions

```
#include <cstdlib>
#include <ctime>
#include <cstring>
#include <cassert>
#include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <queue>
#include <bitset>
#include <sstream>

using namespace std;

#define fore(i, l, r) for(int i = (l); i < (r); ++i)
#define forn(i, n) fore(i, 0, n)
#define fori(i, l, r) fore(i, l, (r) + 1)
#define sz(v) int((v).size())
#define all(v) (v).begin(), (v).end()
#define pb push_back
#define mp make_pair
#define X first
#define Y second

#if ( _WIN32 || __WIN32__ )
    #define LLD "%I64d"
#else
    #define LLD "%lld"
#endif

typedef long long li;
typedef long double ld;
typedef pair<int, int> pt;

template<typename T> T abs(T a) { return a < 0 ? -a : a; }
template<typename T> T sqr(T a) { return a*a; }

const int INF = (int)1e9;
const ld EPS = 1e-9;
const ld PI = 3.14159265358979323846264338327795;

int readInt(int l, int r){
    int x;
    if(scanf("%d", &x) != 1){
        fprintf(stderr, "Expected int in range [%d, %d], but haven't found!", l, r);
        throw;
    }
    if(!(l <= x && x <= r)){
        fprintf(stderr, "Expected int in range [%d, %d], but found %d!", l, r, x);
        throw;
    }
    return x;
}

int up[2], nup[2];

void solve(int tid){
    int n = readInt(1, 100000);
    vector<int> b(n);
    forn(i, n){
        b[i] = readInt(1, 100) - 1;
    }

    forn(i, 2)
        up[i] = 0;

    fore(i, 1, n){
        memset(nup, 0, sizeof nup);
        forn(j, 2){
            forn(k, 2){
                nup[j] = max(nup[j], up[k] + abs((k ? b[i - 1] : 0) - (j ? b[i] : 0)));
            }
        }
        memcpy(up, nup, sizeof nup);
    }
    cout << max(up[0], up[1]) << endl;
}

int main(){
#ifdef ssu1
    assert(freopen("input.txt", "rt", stdin));
```

```
    //assert(freopen("output.txt", "wt", stdout));
#endif

    int T = readInt(1, 20);
    forn(Ti, T){
        solve(Ti);
    }
    return 0;
}
```

Feedback

Was this editorial helpful?

Yes No