USACO NOV07 Problem 'telewire' Analysis

by Spencer Liang & Brian Dean

Let H[i] be the original height of the i-th pole, and let f(n, h) be the minimum cost for n poles with the n-th pole having height h. Then:

```
f(n, h) = min { (H[i]-h)^2 + f(n-1, h') + C|h'-h| } for all h'
```

With a straightforward dynamic programming implementation, this runs in $O(N*H^2)$, where H is the maximum height. However, by splitting up the recurrence relation into two cases, one where h' >= h and one where h' < h, we can rewrite it as:

```
f(n, h) = (H[i]-h)^2 + min { -C*h + min { f(n-1, h')+C*h' } (for h' >= h),
                             C*h + min { f(n-1, h')-C*h' } (for h' < h) }

Define low(n, h)  := min over h' >= h { f(n, h')+C*h' }
   and high(n, h) := min over h' < h  { f(n, h')-C*h' }

Then f(n, h) = (H[i]-h)^2 + min { -C*h+low(n-1, h), C*h+high(n-1, h) }
```

low(n, h) and high(n, h) for all n, h can be computed in O(N*H) time; thus f(n, h) can be computed in O(N*H) time as well. A final implementation detail: an array of size O(N*H) exceeds the the memory limit, but only two "rows" of the DP table are needed at a time, so an array of size 2*H is sufficient. Below is Richard Peng's solution:

```
#include <cstdio>
#define MAXN 110000
#define MAXH 101
int h[MAXN], bes[2][MAXH], ans, huge, bes1, c, n;

inline int sqr (int x){return x*x; }
int main (){
    int i, j, pre, cur;
    freopen ("telewire.in", "r", stdin);
    freopen ("telewire.out", "w", stdout);
    huge = 2100000000;
    scanf ("%d%d", &n, &c);
    for (i = 0; i<n; i++)    scanf ("%d\n", &h[i]);
    for (i = 0; i<MAXH; i++)
        bes[0][i] = (i> = h[0]) ? sqr(h[0]-i) : huge;
    for (i = 1; i<n; i++){
        pre = (i+1)%2;
        cur = i%2;
        for (bes1 = huge, j = 0; j<MAXH; j++){
            bes1 <?= bes[pre][j]-j*c;
            bes[cur][j] = bes1+j*c;
        }
        for (bes1 = huge, j = MAXH-1; j> = 0; j--){
            bes1 <?= bes[pre][j]+j*c;
            bes[cur][j] <?= bes1-j*c;
        }
        for (j = 0; j<MAXH; j++)
            bes[cur][j] = (j> = h[i])? (bes[cur][j] + sqr(j-h[i])) : huge;
    }
    ans = huge;
    for (i = 0; i<MAXH; i++) ans<? = bes[cur][i];
    printf ("%d\n", ans);
}
```