USACO JAN08 Problem 'cowrun' Analysis

by Neal Wu

This is a straightforward dynamic programming (DP) problem. To solve the problem, we want to find, for each k such that 0 <= k <= N, the maximum possible distance Bessie could have run after the first k minutes, if she has a rest factor of 0. (For example, if we can obtain a distance of 14 after 5 minutes with a rest factor of 0, or we can obtain a distance of 15 after 5 minutes with a rest factor of 0, we would always choose the second over the first.) Clearly, the best such value for 0 is 0. Then, for each minute i of the N minutes, we can compute all of the next values possible with the following method:

-First, try to not run during the minute, and see if this produces an improvement. (Thus, check if the best value for i is better than the one for i + 1.)

-Then, for each number k from 1 to M, let Bessie run for exactly k minutes and then rest for k minutes. See if this new value produces a greater value than the best value for i + 2k (which is the number of minutes finished after running for k minutes and resting for another k minutes).

Thus, since we do M updates for each of the N minutes, our total complexity is O(NM). The following is a sample solution:

```
#include <cstdio>
using namespace std;

FILE *fout = fopen ("cowrun.out", "w");
FILE *fin = fopen ("cowrun.in", "r");

const int MAXN = 10005;

int N, M;
int dist [MAXN], best [MAXN];

int main ()
{
    fscanf (fin, "%d %d", &N, &M);

    for (int i = 0; i < N; i++)
        fscanf (fin, "%d", dist + i);
    for (int i = 0; i < N; i++)
    {
// skip the value
        if (best [i] > best [i + 1])
            best [i + 1] = best [i];
        int sum = best [i], pos = i;

        for (int j = 0; j < M && pos < N; j++)
        {
// update each value
            sum += dist [i + j];
            pos += 2;
            if (sum > best [pos])
                best [pos] = sum;
        }
    }
    fprintf (fout, "%d\n", best [N]);
    return 0;
}
```