USACO OPEN11 Problem 'mowlawn' Analysis

by Neal Wu

First compute partial sums, so that psum [i] = $E_0 + ... + E_{i-1}$. Then, if we let dp [i] be the best possible total over all subsets of the first i cows, since we can choose at most K cows in a row we have the following recurrence:

dp [i] = min over all j such that i - K <= j <= i of dp [j - 1] + $E_j + ... + E_{i-1}$ = dp [j - 1] + psum [i] - psum [j].

This immediately gives us an O(NK) solution. However, note that the psum [i] term is independent of j, while dp [j - 1] - psum [j] depends only on j. Thus, we can store paired values of (dp [j - 1] - psum [j], j) in a heap, which allows us to quickly compute the maximum value of dp [j - 1] - psum [j] for j >= i - K in an amortized fashion. (In particular, we have a max heap sorted by dp [j - 1] - psum [j], and each time we perform a query we pop the heap until the index of the top is at least i - K.) This gives us an O(N log N) solution:

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

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

const int MAXN = 100005;

struct data
{
    int ind;
    long long val;

    inline bool operator < (const data &o) const
    {
        return val < o.val;
    }
};

int N, K;
long long psum [MAXN], dp [MAXN];
priority_queue <data> hp;

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

    for (int cow, i = psum [0] = 0; i < N; i++)
    {
        fscanf (fin, "%d", &cow);
        psum [i + 1] = psum [i] + cow;
    }

    hp.push ((data) {-1, 0});
```

```
    for (int i = 0; i <= N; i++)
    {
        while (hp.top ().ind < i - K - 1)
            hp.pop ();

        dp [i] = hp.top ().val + psum [i];
        hp.push ((data) {i, dp [i] - psum [i + 1]});
    }

    fprintf (fout, "%lld\n", dp [N]);
    return 0;
}
```