USACO NOV11 Problem 'median' Analysis

by Brian Dean

If we replace each number less than X with -1 and each number greater than or equal to X with +1, then this problem reduces to counting the number of subarrays with nonnegative sums. There are a few ways to approach this. First, suppose we instead consider the complementary problem of counting the number of subarrays with negative sums. If we precompute an array P of prefix sums (so P[j] gives the sum of A[1...j]), subrrays A[i+1..j] with negative sums correspond to pairs (i,j) with i < j but P[i] > P[j]. Such pairs are called "inversions" in P, and counting inversions is a relatively standard algorithmic problem; this can be done in O(n log n) time, for example, via divide and conquer using a modified merge sort. We can also approach the original problem (counting subarrays with nonnegative sums) from a geometric perspective: build n points (j, P[j]) in the 2D plane, and for each point, we want to count the number of points in its "lower left quadrant" -- points (i, P[i]) with i<j and P[i]<=P[j].

A clever and very concise solution is shown below, due to Nathan Pinsker. Nathan uses a binary index tree -- a useful data structure for many contest problems since it can be coded very quickly. The binary index tree implicitly encodes an array A[1..n] and supports two operations: query(p), which returns the sum of the prefix A[1..p], and update(p), which changes the value of A[p] (in our case here, we only need to increment A[p]). For the geometric problem above, Nathan's code scans the points from left to right and uses the binary index tree to count the number of points in the lower-left quadrant of each point in sequence.

```
#include <cstdio>
using namespace std;
#define MAXN 100005

int n, x, a[MAXN], b[2 * MAXN], s;
long long total;
FILE *in = fopen("median.in", "r"), *out = fopen("median.out", "w");

int query(int p) {
    int t = 0;
    for (int i=(p + MAXN); i; i -= (i & -i))
        t += b[i];
    return t;
}

void update(int p) {
    for (int i=(p + MAXN); i < 2*MAXN; i += (i & -i))
        b[i]++;
}

int main() {
    fscanf(in, "%d%d", &n, &x);
    for (int i=0; i<n; ++i) {
        fscanf(in, "%d", &a[i]);
        a[i] = (a[i] >= x ? 1 : -1);
    }
    update(0);

    for (int i=0; i<n; ++i) {
        s += a[i];
        total += query(s);
        update(s);
    }
    fprintf(out, "%lld\n", total);
}
```