Mock Interviews @ ACM

Solution Card

Pre-rade Flags

Problem Statement

N students numbered 1...N stream through FitzRandolph gate, flags in hand, to have their faces captured by hundreds of waiting cameras along the walkway. One photographer wants a picture where every student in it is holding a Princeton flag. However, a flag shortage has resulted in some students not receiving a flag! Luckily, at the last minute, a box of flags was found, which can be handed out to specific students before they pass the gate. Given that a block of K contiguous students are in the photographer's frame at any point in time, and given the B numbers denoting students without flags, find the minimum number of extra flags we need to give out so that the photographer can take their picture.

Source: http://www.usaco.org/index.php?page=viewproblem2&cpid=715

Hints

Hint 1: How can we efficiently check contiguous blocks of K students? Which blocks are sufficient to check? Answer: blocks where the leftmost student already has a flag.

Follow-ups

- Can we achieve linear time with respect to N? What might be a tradeoff/cost of such an approach?

Solution

We start by making an observation about the optimal blocks of students to consider such that we minimize the number of extra flags. We claim that there is an optimal block where the leftmost student already has a flag. Assume for the sake of contradiction that this is not the case, and consider any optimal block. If we slide it over to the leftmost contiguous block to the right of the optimal one such that the leftmost student has a flag already, note that we removed students that already have flags.

Therefore, in the worst case, this block requires at most the original number of flags to be added, making it optimal.

Therefore, we only need to constrain ourselves to consider contiguous blocks where the leftmost student already has a flag. If we sort the students with flags, we can use binary search to find the rightmost student with a flag that would be within a block of K. This gives us an O(NlogN + BlogN) algorithm. The solution below is actually O(NlogN+B); it leverages the fact that as we iterate over the leftmost student from left to right, the rightmost student that would be valid also iterates from left to right.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

vector<int> A;

int main() {
  int N, K, B;
  cin >> N >> K >> B;

  A.resize(B);
  for (int i = 0; i < B; i++) {
    cin >> A[i];
  }
  sort(A.begin(), A.end());

  int hia = 0;
  while (hia < B && A[hia] <= K) {
    hia++;
  }

  int result = hia;
  for (int i = 0; i < B; i++) {
    if (A[i] + K > N) {
      break;
    }
    while (hia < B && A[hia] <= A[i] + K) {
      hia++;
    }
    result = min(result, hia - i - 1);
  }
  cout << result << endl;
```

```
  return 0;
}
```

An alternative approach, which can get the running time as low as O(N) is to think of the array of students as a binary array: 0 for a student with a flag, 1 for a student missing one. We then want to find a subarray of length K whose sum is minimal, which can be done very easily after first computing an array P of "prefix" sums, where P[j] gives the sum of the first j elements of our binary array. The sum of any range from student i to student j is then easy to obtain by taking P[j]−P[i−1], so evaluating every length-K range takes only O(N) time.

Between these two approaches, note that the first approach can more easily extend to the case where students are at potentially large x coordinates on a number line, not just at locations small enough to fit into an array.

Analysis

First approach

Time: O(NlogN+B)

Space: O(B)

Second approach

Time: O(N)

Space: O(N)