Array Manipulation ☆

AP Editorial by Amit Pandey

You are given a list of size n, initialized with zeroes. You have to perform m queries on the list and output the maximum of final values of all the n elements in the list. For every query, you are given three integers a, b and k and you have to add value k to all the elements ranging from index a to b(both inclusive).

Sub-Optimal Brute Force:

Given each update $a\ b\ k$, for each index in the range from [a, b], add the value k to each number in the range.

The final step is to go through the whole array and find the maximum value and print that maximum value.

The complexity of this solution is O($n \cdot m$) which is too high to pass in time.

Optimal:

- Given a range[a, b] and a value k we need to add k to all the numbers whose indices are in the range from [a, b].
- We can do an O(1) update by adding k to index a and add $-k$ to index $(b+1)$.
- Doing this kind of update, the i^{th} number in the array will be prefix sum of array from index 1 to i because we are adding k to the value at index a and subtracting k from the value at index $b+1$ and taking prefix sum will give us the actual value for each index after m operations .
- So, we can do all m updates in O(m) time. Now we have to check the largest number in the original array. i.e. the index i such that prefix sum attains the maximum value.
- We can calculate all prefix sums as well as maximum prefix sum in O(n) time which will execute in time.

Optimal:

- This can be further optimized to run in O(m logm) time because we have to check the value of prefix sum at only $2 \times m$ indices. i.e. a and b values of all the updates.
- We have, in total m queries and each query has a range [a, b] which needs to be updated. So, in total we have $2 \times m$ indices.
- For each query, we can insert both $a,\ k$ and $b+1,\ -k$ in an array and sort the array.
- Now, we have to just take the prefix sum of the array and find the maximum element which will be our answer.

  Check the setter's code for better understanding.

Note: If you thought of solving it using segment tree with lazy propogation, it won't pass here as n can be as high as 10^7.

Set by amititkgp

Problem Setter's code:

```
#include <bits/stdc++.h>

using namespace std;
long a[400009];

int main() {

    int n;
    int m;
    cin >> n >> m;
    vector < pair < int, int > > v;

    for(int a0 = 0; a0 < m; a0++) {

        int a;
        int b;
        int k;
        cin >> a >> b >> k;

        //storing the query
        //this will add k in the prefix sum for index >= a
        v.push_back(make_pair(a, k));

        //adding -1*k will remove k from the prefix sum for index > b
```

STATISTICS

Difficulty:	Hard
Success Rate:	54.86%
Time Complexity:	O(m logm)
Required Knowledge:	Sorting, Implementation
Publish Date:	Jun 03 2014

Originally featured in Weekly Challenges - Week 4

Of the 1783 contest participants, 658 (36.9%) submitted code for this challenge.

NEED HELP?

View discussions

View top submissions

```
        v.push_back(make_pair(b+1, -1 * k));
    }

    long mx = 0, sum = 0;

    sort(v.begin(), v.end());

    for(int i=0 ; i<2*m; i++) {

        sum += v[i].second;
        mx = max(mx, sum);

    }

    cout<<mx<<endl;
    return 0;
}
```

Tested by jpierce88

Problem Tester's code:

```
def arrayManipulation(n, queries):
    arr = [0] * (n+1)
    # add the value at first index
    # subtract the value at last index + 1
    for q in queries:
        start, end, amt = q
        arr[start-1] += amt
        arr[end] -= amt

    # max value and running sum
    mv = -1
    running = 0
    for a in arr:
        running += a
        if running > mv:
            mv = running

        return mv
```

Feedback

Was this editorial helpful?

Yes No