Saturday, January 26, 2019

Array Manipulation - Hacker Rank Solution

Array Manipulation - Hacker Rank Solution

Starting with a 1-indexed array of zeros and a list of operations, for each operation add a value to each of the array element between two given indices, inclusive. Once all operations have been performed, return the maximum value in your array.

For example, the length of your array of zeros $n = 10$. Your list of queries is as follows:

```
a b k
1 5 3
4 8 7
6 9 1
```

Add the values of k between the indices a and b inclusive:

```
index->  1 2 3  4  5 6 7 8 9 10
 [0,0,0, 0, 0,0,0,0,0, 0]
 [3,3,3, 3, 3,0,0,0,0, 0]
 [3,3,3,10,10,7,7,7,0, 0]
 [3,3,3,10,10,8,8,8,1, 0]
```

The largest value is 10 after all operations are performed.

Function Description

Complete the function *arrayManipulation* in the editor below. It must return an integer, the maximum value in the resulting array.

arrayManipulation has the following parameters:

- *n* - the number of elements in your array
- *queries* - a two dimensional array of queries where each *queries[i]* contains three integers, *a*, *b*, and *k*.

Input Format

The first line contains two space-separated integers n and m, the size of the array and the number of operations.
Each of the next m lines contains three space-separated integers a, b and k, the left index, right index and summand.

Constraints

- $3 \leq n \leq 10^7$
- $1 \leq m \leq 2 * 10^5$
- $1 \leq a \leq b \leq n$
- $0 \leq k \leq 10^9$

BLOG ARCHIVE

January 2019 (9)

CATEGORIES

Advanced Algorithms Arrays
Bash Bit Manipulation C
Closures And Decorators
Data Structures
Dictionaries And Hashmaps
Dynamic Programming
Greedy Algorithms Implementation
Interview Preparation Kit Introducti
Java Linked List Linux Shell
Miscellaneous Python Queues
Recursion And Backtracking Regex
Search Sorting
String Manipulation Trees
Warm-Up Challenges Warmup

POPULAR POSTS

Python If-Else - Hacker Rank Solution
Print Function - Hacker Rank Solution
Sock Merchant - Hacker Rank Solution
Diagonal Difference - Hacker Rank Solu
A Very Big Sum - - Hacker Rank Solutior
Time Conversion - Hacker Rank Solutior
Simple Array Sum - Hacker Rank Solutic
Write a function - Hacker Rank Solution
Nested Lists - Hacker Rank Solution
Lists - Hacker Rank Solution

Output Format

Return the integer maximum value in the finished array.

Sample Input

```
5 3
1 2 100
2 5 100
3 4 100
```

Sample Output

```
200
```

Explanation

After the first update list will be `100 100 0 0 0` .

After the second update list will be `100 200 100 100 100` .

After the third update list will be `100 200 200 200 100` .

The required answer will be 200.

PAGEVIEWS PAST WEEK

Array Manipulation - Hacker Rank Solution

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.

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
      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;
}
```

Problem Tester'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;

      v.push_back(make_pair(a, k));
      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;
       }
```

at January 26, 2019

Labels: Arrays, Interview Preparation Kit

1 Comment:

Anon May 31, 2020 at 9:55 AM

Python 3 Solution function

```
def arrayManipulation(n, queries):
summ, maxx = 0, 0
arr = [0]*(n+1)
for i in queries:
a, b, k = i[0], i[1], i[2]
arr[a-1] += k
arr[b] -= k
arr = arr[:n]
for i in arr:
summ += i
maxx = max(summ, maxx)
return maxx
```

Reply

Enter your comment...

Comment as: Google Accour

Publish Preview

NEWER POST OLDER POST

CONTACT US

Name

Email *

Message *

Send

TAGS

Advanced Algorithms Arrays Bash

Bit Manipulation C

Closures And Decorators Data Structures

Dictionaries And Hashmaps

Dynamic Programming Greedy Algorithms

Implementation Interview Preparation Kit

Introduction Java Linked List

Linux Shell Miscellaneous Python

Get more programming stuff in your inbox instantly by Subscribing to us. So you will g email everytime we post something new h

We guarantee you won't get any other SPAM

Enter email id here... Subscribe Now

Queues Recursion And Backtracking

Regex Search Sorting

String Manipulation Trees

Warm-Up Challenges Warmup