Array Manipulation

Problem | Submissions | Leaderboard | Discussions

Submitted a few seconds ago • Score: 40.00 Status: Accepted

✓ Test Case #0	✓ Test Case #1	✓ Test Case #2
✓ Test Case #3	✓ Test Case #4	✓ Test Case #5
✓ Test Case #6	✓ Test Case #7	✓ Test Case #8
✓ Test Case #9	✓ Test Case #10	✓ Test Case #11
✓ Test Case #12	✓ Test Case #13	✓ Test Case #14
✓ Test Case #15		

Submitted Code

nguage: C++ Open in editor

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

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'arrayManipulation' function below.
 *
 * The function is expected to return a LONG_INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER n
 *  2. 2D_INTEGER_ARRAY queries
 */

long arrayManipulation(int n, vector<vector<int>> queries) {

    int no_of_operations = queries.size(); // finding the number of operations using vector size
    vector<long> final_array(n); // initializing final array

    for (int i=0; i<no_of_operations; i++){ // looping no of operations times
        int start = queries[i][0];
        int end = queries[i][1];
        int adder = queries[i][2];

        // using cumilative some method to reduce time complexity
        final_array[start-1] += adder;
        if (end < n){
            final_array[end] -= adder;
        }
    }

    long max = 0;
    long sum = 0;
    // finding the maximum value
    for (int i=0; i<n; i++){
        sum+=final_array[i];
        if (sum > max){
            max = sum;
        }
    }
    return max;

// we can also use nested for loops but they'll increase time complexity as the input size becomes larger
//     int no_of_operations = queries.size();
//     vector<long> final_array(n,0);

//     for (int i=0; i<no_of_operations; i++){
//         for (int j=queries[i][0]; j<=queries[i][1]; j++){
//             final_array[j-1] += queries[i][2];
//         }
//     }

//     long max = final_array[0];
//     for (int i=0; i<n; i++){
//         if(final_array[i]>max){
//             max = final_array[i];
//         }
//     }

}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string first_multiple_input_temp;
    getline(cin, first_multiple_input_temp);

    vector<string> first_multiple_input = split(rtrim(first_multiple_input_temp));

    int n = stoi(first_multiple_input[0]);

    int m = stoi(first_multiple_input[1]);

    vector<vector<int>> queries(m);

    for (int i = 0; i < m; i++) {
        queries[i].resize(3);

        string queries_row_temp_temp;
        getline(cin, queries_row_temp_temp);

        vector<string> queries_row_temp = split(rtrim(queries_row_temp_temp));

        for (int j = 0; j < 3; j++) {
            int queries_row_item = stoi(queries_row_temp[j]);

            queries[i][j] = queries_row_item;
        }
    }

    long result = arrayManipulation(n, queries);

    fout << result << "\n";

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}

```