L20 : Range Query 1

Github : jitendrabhadoria

1-Tut : Minimum In SubArray

Send Feedback

You are given a sequence A[1], A[2], ..., A[N] of N numbers (0 ≤ A[i] ≤ 10^8 , 2 ≤ N ≤ 10^5). There are Q queries, and each query is defined in either of the following two ways:

Query on range:

You are given two numbers i and j; the answer to each query is the minimum number between the range (i, j) (including both i and j). Note that in this query, i and j lies in the range: [1, N].

Update query:

You are given two numbers i and B; update A[i] to B. Note that in this query, i lies in the range: [1, N].

Input Format:

The first line of input contains two integers: N and Q, representing the length of the sequence and the number of queries.
The second line of input contains N space-separated integers, A[i].
Next, Q lines contain the queries. Each line contains one character, followed by two space-separated integers. For the query on range, the character will be q and for the update query, the character will be u.

Constraints:

1 ≤ i ≤ N
0 ≤ B ≤ 10^8
1 ≤ i <= j ≤ N

Output Format:

For each query on range, print the minimum number between the range, in a new line.

Sample Input 1:

```
4 3
5 2 4 3
q 1 3
u 1 1
q 3 4
```

Sample Output 1:

```
2

3
```

```
1. #include<bits/stdc++.h>
2. using namespace std;
3.
4. void buildtree(int* arr, int* tree, int start, int end, int treenode)
5. {
6.         if (start == end)
7.         {
8.                 tree[treenode] = arr[start];
```

```
9.                      return;
10.         }
11.         int mid = (start + end) / 2;
12.         buildtree(arr, tree, start, mid, 2 * treenode);
13.         buildtree(arr, tree, mid + 1, end, 2 * treenode + 1);
14.         tree[treenode] = min(tree[2 * treenode] , tree[2 * treenode + 1]);
15. }
16. void updatetree(int* arr, int* tree, int start, int end, int treenode, int index, int value)
17. {
18.         if (start == end)
19.         {
20.                 arr[index] = value;
21.                 tree[treenode] = value;
22.                 return;
23.         }
24.         int mid = (start + end) / 2;
25.         if (index > mid)//right
26.         {
27.                 updatetree(arr, tree, mid + 1, end, 2 * treenode + 1, index, value);
28.         }
29.         else//left
30.         {
31.                 updatetree(arr, tree, start, mid, 2 * treenode, index, value);
32.         }
33.         tree[treenode] = min(tree[2 * treenode] , tree[2 * treenode + 1]);
34. }
35. int query(int* tree, int start, int end, int treenode, int left, int right)
36. {
37.         if(start>right || end <left)
38.         {
39.                 return INT_MAX;
40.
41.         }
42.
43.    if(start>=left && end<=right)
44.    {
45.        return tree[treenode];
46.    }
47.         int mid=(start+end)/2;
48.         int ans1=query(tree, start, mid, 2*treenode, left, right);
49.         int ans2=query(tree, mid+1, end, 2*treenode+1, left, right);
50.         return min(ans1, ans2);
51. }
52. int main()
```

```
{
        int n, q;
        cin>>n>>q;
        int *arr=new int [n];
        for(int i=0; i<n; i++)
        {
                cin>>arr[i];
        }
        int *tree=new int [4*n];
        buildtree(arr, tree, 0, n - 1, 1);
        //at this point of time i have my tree with me.
        while(q--)
        {
                char query_type;
    int l,r,x,y;

                cin>>query_type>>l>>r;

                if(query_type=='q')
                {
                        cout<<query(tree, 0, n-1, 1, l-1, r-1)<<endl;
                }
                else
                {
                        x=l;
        y=r;
                        //we need to update arr[x]=y;
                        updatetree(arr, tree, 0, n - 1, 1, x-1, y);
                }
        }
}
```

2-Tut : Maximum Pair Sum

Send Feedback

You are given a sequence A[1], A[2], ..., A[N], (0 ≤ A[i] ≤ 10^8 , 2 ≤ N ≤ 10^5). There are two types of operations and they are defined as follows:

Update:

This will be indicated in the input of a 'U' followed by space and then two integers i and x.

```
U i x
```

This operation sets the value of A[i] to x.

Query:

This will be indicated in the input of a 'Q' followed by a single space and then two integers x and y.

Q x y
You must find two integers i and j such that x ≤ i, j ≤ y and i != j, such that the sum A[i]+A[j] is maximized.

Print the sum A[i]+A[j].

Input Format:

The first line of input contains an integer N, representing the length of the sequence.
The second line of input contains of N space separated integers, A[i].
The third line of input contains an integer Q, Q ≤ 10^5, representing the number of operations.
Next Q lines contain the operations.

Constraints:

1 ≤ i ≤ N
0 ≤ x ≤ 10^8
1 ≤ x < y ≤ N

Output Format:

For each query, print the maximum sum mentioned above, in a new line.

Sample Input 1:

```
5
1 2 3 4 5
6
Q 2 4
Q 2 5
U 1 6
Q 1 5
U 1 7
Q 1 5
```

Sample Output 1:

```
7
9
11

12
```

```
#include<bits/stdc++.h>
using namespace std;

pair<int,int> query(pair<int,int>* tree, int start, int end, int treeNode, int left, int right){

        //Completely out
        if (left>end || right<start)
        {
                pair<int,int> temp = make_pair(INT_MIN, INT_MIN);
                return temp;
        }

        //Completely inside
        if (start>=left && end<=right)
```

```
15.         {
16.                 return tree[treeNode];
17.         }
18.
19.         //Partially inside
20.         int mid = (start+end)/2;
21.
22.         pair<int,int> l = query(tree, start, mid, 2*treeNode+1, left, right);
23.         pair<int,int> r = query(tree, mid+1, end, 2*treeNode+2, left, right);
24.
25.
26.         pair<int,int> result;
27.         result.first = max(l.first,r.first);
28.    result.second = min(max(l.first,r.second),max(l.second,r.first));
29.         return result;
30.
31. }
32.
33.
34. void update(int* arr, pair<int,int>* tree, int start, int end, int treeNode, int idx, int value){
35.         int mid = (start+end)/2;
36.
37.         if (start == end)
38.         {
39.                 arr[idx] = value;
40.                 tree[treeNode] = make_pair(value, INT_MIN);
41.                 return;
42.         }
43.
44.         if (idx<=mid)
45.         {
46.                 update(arr, tree, start, mid, 2*treeNode+1, idx, value);
47.         }else{
48.
49.                 update(arr, tree, mid+1, end, 2*treeNode+2, idx, value);
50.         }
51.
52.         pair<int,int> left = tree[2*treeNode+1];
53.         pair<int,int> right = tree[2*treeNode+2];
54.
55.         tree[treeNode].first = max(left.first,right.first);
56.         tree[treeNode].second =
    min(max(left.first,right.second),max(left.second,right.first));
57.         return;
```

```
58. }
59.
60.
61. void create(int* arr, pair<int,int>* tree, int start, int end, int treeNode){
62.         if (end == start)
63.         {
64.                 tree[treeNode] = make_pair(arr[start], INT_MIN);
65.                 return;
66.         }
67.
68.         int mid = (start+end)/2;
69.
70.         create(arr, tree, start, mid, 2*treeNode+1);
71.         create(arr, tree, mid+1, end, 2*treeNode+2);
72.
73.         pair<int, int> left = tree[2*treeNode+1];
74.         pair<int, int> right = tree[2*treeNode+2];
75.
76.         tree[treeNode].first = max(left.first,right.first);
77.         tree[treeNode].second =
    min(max(left.first,right.second),max(left.second,right.first));
78.
79.         return;
80.
81.
82. }
83.
84. int main()
85. {
86.         int n, q;
87.         cin>>n;
88.
89.         int* arr = new int[n];
90.         for (int i = 0; i < n; ++i)
91.         {
92.                 cin>>arr[i];
93.         }
94.
95.         cin>>q;
96.         pair<int,int>* tree = new pair<int,int>[4*n];
97.         create(arr, tree, 0, n-1, 0);
98.         while(q--){
99.                 char a;
100.                int b, c;
```

```
            cin>>a>>b>>c;

            if (a == 'Q')
            {
                pair<int,int> result = query(tree, 0, n-1, 0, b-1, c-1);
                cout << result.first+result.second << endl;
            }else{
                update(arr, tree, 0, n-1, 0, b-1, c);
            }

        }
        return 0 ;
    }
```

3-Tut : Maximum Sum In Subarray

Send Feedback

You are given a sequence A[1], A[2], ..., A[N].

A **query** is defined as follows:

Query(x,y) = Max { a[i]+a[i+1]+...+a[j] ; x ≤ i ≤ j ≤ y }.
Given M queries, write a program that outputs the results of these queries.

Input Format:

The first line of input contains an integer N.
In the second line contains N space separated integers.
The third line contains the integer M.
Next M lines contains 2 integers x and y.

Output Format:

For each query, print the answer in a new line.

Constraints:

1 <= N <= 10^5
1 < = Q <= 10^5
-10^4 <= arr[i] <= 10^4

Sample Input 1:

3
-1 2 3
1
1 2

Sample Output 1:

2

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. struct attrs {
```

```
4.          int max_sum;
5.          int sum;
6.          int best_prefix_sum;
7.          int best_suffix_sum;
8.  };
9.  void build_tree(int* arr, attrs* tree, int start, int end, int treenode)
10. {
11.         if (start == end)
12.         {
13.                 tree[treenode].sum = arr[start];
14.                 tree[treenode].max_sum = arr[start];
15.                 tree[treenode].best_suffix_sum = arr[start];
16.                 tree[treenode].best_prefix_sum = arr[start];
17.                 return;
18.         }
19.         int mid = (start + end) / 2;
20.         build_tree(arr, tree, start, mid, 2 * treenode);
21.         build_tree(arr, tree, mid + 1, end, 2 * treenode + 1);
22.         tree[treenode].sum = tree[2 * treenode].sum + tree[2 * treenode + 1].sum;
23.         tree[treenode].best_prefix_sum = max(tree[2 * treenode].best_prefix_sum, tree[2
    * treenode].sum + tree[2 * treenode + 1].best_prefix_sum);
24.         tree[treenode].best_suffix_sum = max(tree[2 * treenode + 1].best_suffix_sum,
    tree[2 * treenode + 1].sum + tree[2 * treenode].best_suffix_sum);
25.         tree[treenode].max_sum = max
26.         (
27.                 tree[2 * treenode].max_sum,
28.                 max
29.                 (
30.                         tree[2 * treenode + 1].max_sum,
31.                         max
32.                         (
33.                                 tree[2 * treenode].sum + tree[2 * treenode +
    1].best_prefix_sum,
34.                                 max
35.                                 (
36.                                         tree[2 * treenode + 1].sum + tree[2 *
    treenode].best_suffix_sum,
37.                                         tree[2 * treenode].best_suffix_sum + tree[2 *
    treenode + 1].best_prefix_sum
38.                                 )
39.                         )
40.                 )
41.         );
42. }
```

```
attrs query(attrs* tree, int start, int end, int treenode, int left, int right)
{
	//completely outside
	if (start > right || end < left)
	{
		return {-100000, -100000, -100000, -100000};
	//not used int min here because during recursion it will become +ve if any number is subtracted from it.
	}
	//completely inside
	if (start >= left && end <= right)
	{
		return tree[treenode];
	}
	//partially outside and partially inside
	int mid = (start + end) / 2;
	attrs q1 = query(tree, start, mid, 2 * treenode, left, right);
	attrs q2 = query(tree, mid + 1, end, 2 * treenode + 1, left, right);
	attrs ans;
	ans.sum = q1.sum + q2.sum;
	ans.best_prefix_sum = max(q1.best_prefix_sum, q1.sum + q2.best_prefix_sum);
	ans.best_suffix_sum = max(q1.best_suffix_sum + q2.sum, q2.best_suffix_sum);
	ans.max_sum = max
	(
		q1.max_sum,
		max
		(
			q2.max_sum,
			max
			(
				q1.sum + q2.best_prefix_sum,
				max
				(
					q2.sum + q1.best_suffix_sum,
					q1.best_suffix_sum + q2.best_prefix_sum
				)
			)
		)
	);
	return ans;
}
int main()
{
	int n;
```

```
        cin >> n;
        int* arr = new int[n];
        attrs* tree = new attrs[4 * n];
        for (int i = 0; i < n; i++)
        {
                cin >> arr[i];
        }
        build_tree(arr, tree, 0, n - 1, 1);
        int m;
        cin >> m;
        while (m--)
        {
                int xi, yi;
                cin >> xi >> yi;
                cout << query(tree, 0, n - 1, 1, xi-1, yi-1).max_sum << endl;

        }
    }
```

4-Tut : Sum Of Squares

Send Feedback

Segment trees are extremely useful. In particular "Lazy Propagation" (i.e. see here, for example) allows one to compute sums over a range in O(lg(n)), and update ranges in O(lg(n)) as well. In this problem you will compute something much harder:

The sum of squares over a range with range updates of 2 types:

1) increment in a range

2) set all numbers the same in a range.

There will be T test cases in the input file. First line of the input contains two positive integers, N and Q. The next line contains N integers, each at most 1000. Each of the next Q lines starts with a number, which indicates the type of operation:

2 st nd -- return the sum of the squares of the numbers with indices in [st, nd] {i.e., from st to nd inclusive} (1 <= st <= nd <= N).

1 st nd x -- add "x" to all numbers with indices in [st, nd] (1 <= st <= nd <= N, and 1 <= x <= 1,000).

0 st nd x -- set all numbers with indices in [st, nd] to "x" (1 <= st <= nd <= N, and 1 <= x <= 1,000).

Input Format:

First line of input will contain T(number of test case), each test case follows as.
Line 1: contain two space-separated integers denoting the value of N and Q respectively
Line 2: contain N space-separated integers denoting th value of array elements
Next Q line contain the space separated value for queries as defined as above.

Constraints:

1 <= T <= 10
1 <= N <= 10000
1 <= arr[i] <= 1000
1 <= Q <= 10000

Output Format:

For each test case, print the answer in new line for query of type 2

Sample Input 1:

```
2
4 5
1 2 3 4
2 1 4
0 3 4 1
2 1 4
1 3 4 1
2 1 4
1 1
1
2 1 1
```

Sample Output 1:

```
30
7
13

1
```

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

int t;
int n;

ll tree[400000];
ll arr[100000];

void build(int node, int start,  int end){
    if(start > end)
        return;
    if(start==end){
        //Leaf node.
        tree[node] = arr[start]*arr[start];
        return;
    }
    //Recurse on the left child.
```

```
    int mid = (start+end)>>1;
    int left = node<<1, right = left+1;
    build(left, start, mid);
    //Recurse on the right child.
    build(right, mid+1, end);
    //Internal node will have the sum of both its children.
    tree[node] = tree[left] + tree[right];
}


//Time Complexity O(log(n))
ll query(int node, int start, int end, int l, int r){
    if(start > end || start > r || end < l)
        return 0;
        //complete overlap
    if(l <= start && end <= r)
        return tree[node];
    //partial overlap
    int mid = (start+end)/2;
    int left = node<<1, right = left+1;
        return query(left, start, mid, l, r) + query(right, mid+1, end, l, r);
}

//Worst case complexity is O(n)
void update1(int node, int start, int end, int l, int r, int value){
    //increase values from l to r by value
    //no overlap of the l to r segment on the current segment
    if(start > end || start > r || end < l)
        return;
    if(start==end){
        //Leaf node
        arr[start]+=value;
        tree[node] = arr[start]*arr[start];
        return;
    }
    int mid = (start+end)/2;
    int left = node<<1, right = left+1;
    update1(left, start, mid, l, r, value);
    update1(right, mid+1, end, l, r, value);
    tree[node] = tree[left] + tree[right];
}

void update2(int node, int start, int end, int l, int r, int value){
    //increase values from l to r by value
```

```
    //no overlap of the l to r segment on the current segment
    if(start > end || start > r || end < l)
        return;
    if(start==end){
        //Leaf node
        arr[start] = value;
        tree[node] = arr[start]*arr[start];
        return;
    }
    int mid = (start+end)/2;
    int left = node<<1, right = left+1;
    update2(left, start, mid, l, r, value);
    update2(right, mid+1, end, l, r, value);
    tree[node] = tree[left] + tree[right];
}

int main(){

    cin >> t;
    while(t--){
        memset(arr, 0, sizeof(arr));
        memset(tree, 0, sizeof(tree));
        cin >> n;
        int q;
        cin >> q;
        for(int i = 0;i < n; i++){
            cin >> arr[i];
        }
        build(1, 0, n-1);
        int flag = 0;
        while(q--){
            int type;
            cin >> type;
            if(type == 0){
                int st, end, x;
                    cin >> st >> end >> x;
                    update2(1, 0, n-1, st-1, end-1, x);
                } else if(type == 1){
                    int st, end, x;
                    cin >> st >> end >> x;
                    update1(1, 0, n-1, st-1, end-1, x);
                } else {
                    int st, end;
                    cin >> st >> end;
```

```
                if(flag == 0){
                    flag = 1;
                }
                cout << query(1, 0, n-1, st-1, end-1) << endl;
            }
        }
    }
    return 0;
}
```

5-Ass : Maximum Query

Send Feedback

You are given an array of integet of size N and Q queries in form of (l, r). You are supposed to find the maximum value of array between index l and r (both inclusive)

Input Format:

First line of input contain an integer N (number of elements in the array)
Second line contain N space-separated integers denoting the elements of the array
Third line contain an integer Q (number of queries to be processed)
Next Q line contain two space-separated integers denoting l and r.

Output Format:

For each test case print the output in newline.

Constraints:

1 <= N <= 10^4
1 <= Q <= 10^6
1 <= arr[i] <= 10^9
0 <= l <= r < N

Sample Input:

```
5
1 2 3 5 4
2
0 1
3 4
```

Sample Output:

```
2

5
```

```
#include<bits/stdc++.h>
using namespace std;

int query(int* tree, int start, int end, int treeNode, int left, int right){

        //Completely out
        if (left>end || right<start)
        {
```

```
9.                      int temp = INT_MIN;
10.                     return temp;
11.             }
12.
13.             //Completely inside
14.             if (start>=left && end<=right)
15.             {
16.                     return tree[treeNode];
17.             }
18.
19.             //Partially inside
20.             int mid = (start+end)/2;
21.
22.             int l = query(tree, start, mid, 2*treeNode+1, left, right);
23.             int r = query(tree, mid+1, end, 2*treeNode+2, left, right);
24.
25.
26.             int result;
27.             result = max(l,r);
28.
29.             return result;
30.
31. }
32.
33. void create(int* arr, int* tree, int start, int end, int treeNode){
34.
35.             if (end == start)
36.             {
37.                     tree[treeNode] = arr[start];
38.                     return;
39.             }
40.
41.             int mid = (start+end)/2;
42.
43.             create(arr, tree, start, mid, 2*treeNode+1);
44.             create(arr, tree, mid+1, end, 2*treeNode+2);
45.
46.             int left = tree[2*treeNode+1];
47.             int right = tree[2*treeNode+2];
48.
49.             tree[treeNode] = max(left,right);
50.
51.             return;
52.
```

```
}

int main()
{
        int n, q;
        cin>>n;

        int* arr = new int[n];
        for (int i = 0; i < n; ++i)
        {
                cin>>arr[i];
        }

        cin>>q;
        int* tree = new int[4*n];
        create(arr, tree, 0, n-1, 0);

        while(q--){

                int b, c;
                cin>>b>>c;
                int result = query(tree, 0, n-1, 0, b, c);

    cout << result << endl;

        }

        return 0 ;

}
```

6-Ass : Counting Even/Odd

Send Feedback

Tanmay and Rohit are best buddies. One day Tanmay gives Rohit a problem to test his intelligence and skills. He gives him an array of N natural numbers and asks him to solve the following queries:-

Query 0:

`0 x y`

This operation modifies the element present at index i to x.

Query 1:

`1 x y`

This operation counts the number of even numbers in range l to r inclusive.

Query 2:

2 x y
This operation counts the number of odd numbers in range l to r inclusive.

Input Format:

First line of the input contains the number N.
Next line contains N natural numbers.
Next line contains an integer Q followed by Q queries.
0 x y - modify the number at index x to y.
1 x y - count the number of even numbers in range l to r inclusive.
2 x y - count the number of odd numbers in range l to r inclusive.

Constraints:

1<=N,Q<=10^5
1<=l<=r<=N
0<=Ai<=10^9
1<=x<=N
0<=y<=10^9

Output Format:

For each query, print the answer in a new line.

Note: Indexing starts from 1

Sample Input 1:

```
6
1 2 3 4 5 6
4
1 2 5
2 1 4
0 5 4
1 1 6
```

Sample Output 1:

```
2
2

4
```

```
#include<bits/stdc++.h>
using namespace std;
struct tree_attrs
{
        int even;
        int odd;
};
void build_tree(int* arr, tree_attrs* tree, int start, int end, int treenode)
{
        if (start == end)
        {
```

```
12.                     if (arr[start] % 2 == 0)
13.                     {
14.                             tree[treenode].even = 1;
15.                             tree[treenode].odd = 0;
16.                     }
17.                     else
18.                     {
19.                             tree[treenode].odd = 1;
20.                             tree[treenode].even = 0;
21.                     }
22.                     return;
23.             }
24.             int mid = (start + end) / 2;
25.             build_tree(arr, tree, start, mid, 2 * treenode);
26.             build_tree(arr, tree, mid+1, end, 2 * treenode+1);
27.             tree[treenode].odd = tree[2 * treenode].odd + tree[2 * treenode + 1].odd;
28.             tree[treenode].even = tree[2 * treenode].even + tree[2 * treenode + 1].even;
29. }
30. void update_modify(int *arr, tree_attrs* tree, int start, int end, int treenode, int index, int
    value)
31. {
32.             if (start == end)
33.             {
34.                     arr[index] = value;
35.                     if (value % 2 == 0)
36.                     {
37.                             tree[treenode].even = 1;
38.                             tree[treenode].odd = 0;
39.                     }
40.                     else
41.                     {
42.                             tree[treenode].even = 0;
43.                             tree[treenode].odd = 1;
44.                     }
45.                     return;
46.             }
47.             int mid = (start + end) / 2;
48.             if (index > mid)//right
49.             {
50.                     update_modify(arr, tree, mid + 1, end, 2 * treenode + 1, index, value);
51.             }
52.             else//left
53.             {
54.                     update_modify(arr, tree, start, mid, 2 * treenode, index, value);
```

```
55.         }
56.         tree[treenode].even = tree[2 * treenode].even + tree[2 * treenode + 1].even;
57.         tree[treenode].odd = tree[2 * treenode].odd + tree[2 * treenode + 1].odd;
58. }
59. tree_attrs query(tree_attrs* tree, int start, int end, int treenode, int left, int right)
60. {
61.         //completely outside
62.         if (start > right || end < left)
63.         {
64.                 tree_attrs ans;
65.                 ans.even = 0;
66.                 ans.odd = 0;
67.                 return ans;
68.         }
69.         //complete overlap
70.         if (start >= left && end <= right)
71.         {
72.                 return tree[treenode];
73.         }
74.         //partial overlap
75.         int mid = (start + end) / 2;
76.         tree_attrs left_child = query(tree, start, mid, 2 * treenode, left, right);
77.         tree_attrs right_child = query(tree, mid + 1, end, 2 * treenode + 1, left, right);
78.         tree_attrs ans;
79.         ans.even = left_child.even + right_child.even;
80.         ans.odd = left_child.odd + right_child.odd;
81.         return ans;
82. }
83. int main()
84. {
85.         int n;
86.         cin >> n;
87.         int* arr = new int[n];
88.         for (int i = 0; i < n; i++)
89.         {
90.                 cin >> arr[i];
91.         }
92.         tree_attrs* tree = new tree_attrs[4 * n]();
93.         build_tree(arr, tree, 0, n - 1, 1);
94.         int q;
95.         cin >> q;
96.         while (q--)
97.         {
98.                 int query_type;
```

```
99.                 cin >> query_type;
100.                if (query_type == 0)
101.                {
102.                        int index, value;
103.                        cin >> index >> value;
104.                        update_modify(arr, tree, 0, n - 1, 1, index-1, value);
105.                }
106.                else if (query_type == 1)
107.                {
108.                        int left, right;
109.                        cin >> left >> right;
110.                        cout << query(tree, 0, n - 1, 1, left - 1, right - 1).even<<endl;
111.                }
112.                else
113.                {
114.                        int left, right;
115.                        cin >> left >> right;
116.                        cout << query(tree, 0, n - 1, 1, left - 1, right - 1).odd<<endl;
117.                }
118.        }
119.    }
```

7-Ass : This is Sparta!

Send Feedback

King Leonidas of Sparta is preparing his men and country for a war against the Persian King Xerxes. He has N soldiers with him and he has arranged them in a line at The Hot Gates. Let us number them from 1 to N. Leonidas will fight Xerxes' army for Q days, and each day he can send only one of his men to fight. For each warrior, we know 2 traits: Strength and Cowardice. These are given to us in a form of integer. Each day, Leonidas can choose his warrior from a range Li to Ri, and he will choose the warrior with maximum Strength value. If there is more than one warrior having the same maximum Strength value, he will choose the warrior with minimum Cowardice value. If there is still more than 1 warrior with the same maximum Strength value and same minimum Cowardice value, he chooses the one with lower index in line.

King Leonidas is ready to lay his life for Sparta. You, his right hand man, have to help him save Sparta by helping him choose a warrior for each day.

Input Format:

First line contains a single integer N, denoting the number of warriors Leonidas has.
Second line contains N space separated integers, representing Strength of ith warrior.
Third line contains N space separated integers, representing Cowardice of ith warrior
Next line contains a single integer Q, denoting the number of days Queen Vasya chooses a warrior.
Each of the next Q lines contains 2 integers Li and Ri.

Constraints:

1 ≤ N,Q ≤ 10^5
1 ≤ Ai,Bi ≤ 10^9
1 ≤ Li ≤ Ri

Output Format:

For each Li and Ri, print the index of the warrior that King Leonidas should choose.

Sample Input 1:

```
5
1 8 4 6 8
4 8 6 3 7
4
1 4
2 4
3 4
1 5
```

Sample Output 1:

```
2
2
4

5
```

```
#include<bits/stdc++.h>
using namespace std;
struct warriors
{
        int a;
        int b;
};
struct tree_attrs
{
        int a;
        int b;
        int index;
};
void build_tree(warriors* arr, tree_attrs* tree, int start, int end, int treenode)
{
        if (start == end)
        {
                tree[treenode].a = arr[start].a;
                tree[treenode].b = arr[start].b;
                tree[treenode].index = start;
                return;
        }
        int mid = (start + end) / 2;
        build_tree(arr, tree, start, mid, 2 * treenode);
```

```
25.         build_tree(arr, tree, mid + 1, end, 2 * treenode + 1);
26.         tree_attrs left_child = tree[2 * treenode];
27.         tree_attrs right_child = tree[2 * treenode + 1];
28.         if (left_child.a > right_child.a)
29.         {
30.                 tree[treenode].a = left_child.a;
31.                 tree[treenode].b = left_child.b;
32.                 tree[treenode].index = left_child.index;
33.         }
34.         else if(left_child.a < right_child.a)
35.         {
36.                 tree[treenode].a = right_child.a;
37.                 tree[treenode].b = right_child.b;
38.                 tree[treenode].index = right_child.index;
39.         }
40.         else
41.         {
42.                 if (left_child.b < right_child.b)
43.                 {
44.                         tree[treenode].a = left_child.a;
45.                         tree[treenode].b = left_child.b;
46.                         tree[treenode].index = left_child.index;
47.                 }
48.                 else if (left_child.b > right_child.b)
49.                 {
50.                         tree[treenode].a = right_child.a;
51.                         tree[treenode].b = right_child.b;
52.                         tree[treenode].index = right_child.index;
53.                 }
54.                 else
55.                 {
56.                         if (left_child.index < right_child.index)
57.                         {
58.                                 tree[treenode].a = left_child.a;
59.                                 tree[treenode].b = left_child.b;
60.                                 tree[treenode].index = left_child.index;
61.                         }
62.                         else
63.                         {
64.                                 tree[treenode].a = right_child.a;
65.                                 tree[treenode].b = right_child.b;
66.                                 tree[treenode].index = right_child.index;
67.                         }
68.                 }
```

```
69.         }
70. }
71. tree_attrs query(tree_attrs* tree, int start, int end, int treenode, int left, int right)
72. {
73.         //completely outside
74.         if (start > right || end < left)
75.         {
76.                 tree_attrs ans;
77.                 ans.a = INT_MIN;
78.                 ans.b = INT_MAX;
79.                 ans.index = INT_MAX;
80.                 return ans;
81.         }
82.         //completely inside
83.         if (start >= left && end <= right)
84.         {
85.                 return tree[treenode];
86.         }
87.         //partial overlap
88.         int mid = (start + end) / 2;
89.         tree_attrs left_child = query(tree, start, mid, 2 * treenode, left, right);
90.         tree_attrs right_child = query(tree, mid + 1, end, 2 * treenode + 1, left, right);
91.         if (left_child.a > right_child.a)
92.         {
93.                 return left_child;
94.         }
95.         else if (left_child.a < right_child.a)
96.         {
97.                 return right_child;
98.         }
99.         else
100.        {
101.                if (left_child.b < right_child.b)
102.                {
103.                        return left_child;
104.                }
105.                else if (left_child.b > right_child.b)
106.                {
107.                        return right_child;
108.                }
109.                else
110.                {
111.                        if (left_child.index < right_child.index)
112.                        {
```

```
                return left_child;
            }
            else
            {
                return right_child;
            }
        }
    }
}
int main()
{
    int n;
    cin >> n;
    warriors* arr = new warriors[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i].a;
    }
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i].b;
    }
    tree_attrs* tree = new tree_attrs[4 * n]();
    build_tree(arr, tree, 0, n - 1, 1);
    int q;
    cin >> q;
    while (q--)
    {
        int left, right;
        cin >> left >> right;
        cout << query(tree, 0, n - 1, 1, left - 1, right - 1).index+1 << endl;
      // I have added 1 because indexing starts from 1
    }
}
```

8-Ass : 2 vs 3

Send Feedback

The fight for the best number in the globe is going to finally come to an end.The top two contenders for the best number are number 2 and number 3.It's the final the entire world was waiting for. Expectorates from all across the globe came to witness the breath taking finals.

The finals began in an astonishing way.A common problem was set for both of them which included both these number.The problem goes like this.

Given a binary string (that is a string consisting of only 0 and 1). They were supposed to perform two types of query on the string.

Type 0: Given two indices l and r.Print the value of the binary string from l to r modulo 3.

Type 1: Given an index l flip the value of that index if and only if the value at that index is 0.

The problem proved to be a really tough one for both of them.Hours passed by but neither of them could solve the problem.So both of them wants you to solve this problem and then you get the right to choose the best number in the globe.

Input format:

The first line contains N denoting the length of the binary string .
The second line contains the N length binary string.Third line contains the integer Q indicating the number of queries to perform.
This is followed up by Q lines where each line contains a query.

Output format:

For each query of Type 0 print the value modulo 3.

Constraints:

1<= N <=10^5
1<= Q <= 10^5
0 <= l <= r < N

Sample Input

```
5
10010
6
0 2 4
0 2 3
1 1
0 0 4
1 1
0 0 3
```

Sample Output

2
1
2

1

```
#include<bits/stdc++.h>
using namespace std;
int power[100001];

void buildPower(){
    power[0] = 1;
    for(int i = 1; i < 100001; i++) power[i] = (power[i-1]*2)%3;
}
void build_tree(int* arr, int* tree, int start, int end, int treenode)
{
        if (start == end)
        {
                tree[treenode] = arr[start];
                return;
        }
        int mid = (start + end) / 2;
        build_tree(arr, tree, start, mid, 2 * treenode);
        build_tree(arr, tree, mid + 1, end, 2 * treenode + 1);
        int right_child = tree[2 * treenode + 1];
        int left_child = tree[2 * treenode];
        tree[treenode] = (power[end-mid]*left_child + right_child)%3;
}

void update_flip(int *arr, int* tree, int start, int end, int treenode, int index)
{
        if (start == end)
        {
                arr[index]=1;
      tree[treenode]=1;
      return;
   }
        int mid = (start + end) / 2;
        if (index > mid)
        {
                update_flip(arr, tree, mid + 1, end, 2 * treenode + 1, index);
        }
        else
        {
                update_flip(arr, tree, start, mid, 2 * treenode, index);
```

```
        }

        int right_child = tree[2 * treenode + 1];
        int left_child = tree[2 * treenode];
        tree[treenode] = (power[end-mid]*left_child + right_child)%3;
}

int query_value(int* tree, int start, int end, int treenode, int left, int right)
{
        //completely outside
        if (start > right || end < left)
        {
                return 0;
        }
        //completely inside
        if (start >= left && end <= right)
        {
                return (tree[treenode]*power[right-end])%3;
        }
        //partial overlap
        int mid = (start + end) / 2;
        int answer_from_left = query_value(tree, start, mid, 2 * treenode, left, right);
        int answer_from_right = query_value(tree, mid + 1, end, 2 * treenode + 1, left,
right);
        return (answer_from_left+answer_from_right)%3;
}
int main()
{
    buildPower();
        int n;
        cin >> n;
        string s;
        cin >> s;
        int* arr = new int[n];
        for (int i = 0; i < n; i++)
        {
                arr[i] = s[i]-'0';
        }
        int* tree = new int[4 * n]();
        build_tree(arr, tree, 0, n - 1, 1);
        int q;
        cin >> q;
        while (q--)
        {
```

```
                int query_type;
                cin >> query_type;
                if (query_type == 0)
                {
                        int left, right;
                        cin >> left >> right;
                        cout << query_value(tree, 0, n - 1, 1, left, right) << endl;;
                }
                else
                {
                        int index;
                        cin >> index;
        if(arr[index]==0)
                            update_flip(arr, tree, 0, n - 1, 1, index);
                }
        }
}
```

9-Ass : Legion of Doom

Send Feedback

Lex Luthor's Legion of Doom is a tough organization to get into, even for greatest supervillains. Recently, a spot has opened up because The Mad Hatter has retired. Harley Quinn doesn't want to waste this opportunity, and jumps at the chance of the interview. But she has a PhD in psychology, not in Computer Science. She has kidnapped you and will let you go only if you are able to solve the evil questions of Lex Luthor.

You are given an array of N elements, which are initially all 0. After that you will be given C commands. They are -

0 p q v - you have to add v to all numbers in the range of p to q (inclusive), where p and q are two indexes of the array.

1 p q - output a line containing a single integer which is the sum of all the array elements between p and q (inclusive)

Input Format:

In the first line you'll be given T, number of test cases.

Each test case will start with N and C. After that you'll be given C commands in the format as mentioned above

Constraints:

1 <= T <= 10

1 <= N, C <= 10000

1 <= val <= 10^8
1 <= p <= q <= N

Output Format:
Print the answers of the queries in new line for each test case.

Sample Input 1:
```
1
8 6
0 2 4 26
0 4 8 80
0 4 5 20
1 8 8
0 5 7 14
1 4 8
```

Sample Output 1:
```
80

508
```

```
#include<bits/stdc++.h>
using namespace std;
typedef long long int ll;
void update_add(ll* tree, ll* lazy, ll start, ll end, ll treenode, ll left, ll right, ll value)
{
        if (start > end)
        {
                return;
        }
        if (lazy[treenode] != 0)
        {
                tree[treenode] += (end - start + 1) * lazy[treenode];
                if (start != end)
                {
                        lazy[2 * treenode] += lazy[treenode];
                        lazy[2 * treenode + 1] += lazy[treenode];
                }
                lazy[treenode] = 0;
        }
        //completely outside
```

```
21.         if (start > right || end < left)
22.         {
23.                 return;
24.         }
25.         //completely inside
26.         if (start >= left && end <= right)
27.         {
28.                 tree[treenode] += (end - start + 1) * value;
29.                 if (start != end)
30.                 {
31.                         lazy[2 * treenode] += value;
32.                         lazy[2 * treenode + 1] += value;
33.                 }
34.                 return;
35.         }
36.         //partial overlap
37.         ll mid = (start + end) / 2;
38.         update_add(tree, lazy, start, mid, 2 * treenode, left, right, value);
39.         update_add(tree, lazy, mid + 1, end, 2 * treenode + 1, left, right, value);
40.         tree[treenode] = tree[2 * treenode] + tree[2 * treenode + 1];
41.         return;
42. }
43. ll query_sum(ll* tree, ll* lazy, ll start, ll end, ll treenode, ll left, ll right)
44. {
45.         if (start > end)
46.         {
47.                 return 0;
48.         }
49.         if (lazy[treenode] != 0)
50.         {
51.                 tree[treenode] += (end - start + 1) * lazy[treenode];
52.                 if (start != end)
53.                 {
54.                         lazy[2 * treenode] += lazy[treenode];
55.                         lazy[2 * treenode + 1] += lazy[treenode];
56.                 }
57.                 lazy[treenode] = 0;
58.         }
59.         //completely outside
60.         if (start > right || end < left)
61.         {
62.                 return 0;
63.         }
64.         //completely inside
```

```
65.         if (start >= left && end <= right)
66.         {
67.                 return tree[treenode];
68.         }
69.         //partial overlap
70.         ll mid = (start + end) / 2;
71.         ll left_child = query_sum(tree, lazy, start, mid, 2 * treenode, left, right);
72.         ll right_child = query_sum(tree, lazy, mid + 1, end, 2 * treenode + 1, left, right);
73.         return left_child + right_child;
74. }
75. int main()
76. {
77.         ll t;
78.         cin >> t;
79.         while (t--)
80.         {
81.                 ll n, c;
82.                 cin >> n >> c;
83.
84.                 ll* tree = new ll[4 * n]();
85.                 ll* lazy = new ll[4 * n]();
86.                 while (c--)
87.                 {
88.                         ll command_type;
89.                         cin >> command_type;
90.                         if (command_type == 0)
91.                         {
92.                                 ll left, right, value;
93.                                 cin >> left >> right >> value;
94.                                 update_add(tree, lazy, 0, n - 1, 1, left - 1, right - 1, value);
95.                         }
96.                         else
97.                         {
98.                                 ll left, right;
99.                                 cin >> left >> right;
100.                                cout << query_sum(tree, lazy, 0, n - 1, 1, left - 1, right - 1)
    << endl;
101.                        }
102.                }
103.        }
104.    }
```

10-Ass : The GCD Dillema

Send Feedback

Dwight is always bragging about how amazing he is at solving complicated problems with much ease. Jim got tired of this and gave him an interesting problem to solve.

Jim gave Dwight a sequence of integers $a_1, a_2, ..., a_n$ and q queries $x_1, x_2, ..., x_q$ on it. For each query x_i Dwight has to count the number of pairs (l, r) such that $1 \le l \le r \le n$ and $GCD(a_l, a_{l+1}, ..., a_r) = x_i$. Dwight is feeling out of his depth here and asked you to be his Secret Assistant to the Regional Manager. Your first task is to help him solve the problem. Are you up to it?

Input Format:

First line of input contains an integer N, representing the number of elements in the sequence.
Second line contains N space-separated integers denoting the elements of the sequence.
Third line of input contains an integer Q, representing the number of queries.
Next Q line contains an integer X.

Constraints:

1 < = N <= 10^4
1 <= arr[i] <= 10^9
1 <= Q <= 10^4
1 <= X <= 10^9

Output Format:

For each query, print the answer in a new line.

Sample Input:

```
2
8 12
3
8
12
4
```

Sample Output:

```
1
1

1
```

```
1. #include<bits/stdc++.h>
2. using namespace std;
3.
4. const int LIM = 100003;
5. int A[LIM], N;
6. map<int, long long> countg[2], finalc; // final[i] stores the countg of subarrays with gcd = i
7. map<int, long long>::iterator it;
8.
9. void precompute() {
10.  // at any state i, countg[prev] stores the gcd's of subarrays ending at A[i - 1] with their
     countg,
```

```
11.  // for making countg[curr], we take the gcd of A[i] with all elements in countg[prev] and add the countg
12.  // original solution by Rachit http://codeforces.com/contest/475/submission/8093857
13.  countg[0][A[0]] = 1;
14.  finalc[A[0]] = 1LL;
15.  int g, curr, prev;
16.  for(int i = 1; i < N; i++) {
17.    curr = i & 1, prev = curr ^ 1;
18.    countg[curr].clear();
19.    countg[curr][A[i]] = 1LL;    // 1 for the subarray containing only A[i], rest will come from taking gcd with those till A[i - 1] in countg[prev]
20.    for(it = countg[prev].begin(); it != countg[prev].end(); it++) {
21.      g = __gcd(it->first, A[i]);
22.      countg[curr][g] += it->second;   // every subarray giving it->first till A[i - 1] will give g with A[i]
23.    }
24.    for(it = countg[curr].begin(); it != countg[curr].end(); it++) finalc[it->first] += it->second;
25.  }
26. }
27.
28. int main() {
29.
30.  cin >> N;
31.  for(int i = 0; i < N; i++) cin >> A[i];
32.  precompute();
33.  int Q, x;
34.  long long ans;
35.  cin >> Q;
36.  while(Q--) {
37.    cin >> x;
38.    ans = finalc[x];
39.    cout << ans << "\n";
40.  }
41.
42.  return 0;
43. }
```

11-Ass : Sheldon and Trains

Send Feedback

Sheldon always tells people, “When you have only one day to visit Los Angeles, make it a Train Day”. He loves spending time while travelling in trains and considers it a fun activity. Sheldon’s mom has come to visit him and he decides to take her out on a train tour of the city of Pasadena, along with his friend Howard. There are n train stations in the city. Howard knows how irritating Sheldon can be during a train ride. So, to keep him busy, Howard gives Sheldon a problem so interesting that he just cannot do

anything else other than devote his entire mind to solving it. The problem goes like this. At the i-th station it's possible to buy only tickets to stations from i + 1 to ai (inclusive). No tickets are sold at the last station. Let $\rho_{i,j}$ be the minimum number of tickets one needs to buy in order to get from stations i to station j. Sheldon's task is to compute the sum of all values $\rho_{i,j}$ among all pairs $1 \le i < j \le n$. As brilliant as he may be, he asked for your help.

Input Format:

First line of input will contain N number of trains
Second line will contain N-1 space-separated integers denoting the values of ai

Output Format:

Print the answer as mentioned above

Constraints:

2 < = N <= 10^5
i + 1 <=arr[i] <= N

Sample Input 1:

7
2 7 5 7 6 7

Sample Output 1:

29

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 111111

struct Node{
    int mmm,idx;
    Node():mmm(0){}
}tree[MAXN<<2];
int x,y,N;
void update(int i,int j,int k){
    if(i==j){
        tree[k].mmm=y;
        tree[k].idx=i;
        return;
    }
    int mid=i+j>>1;
    if(x<=mid) update(i,mid,k<<1);
    else update(mid+1,j,k<<1|1);
    if(tree[k<<1].mmm>tree[k<<1|1].mmm){
        tree[k]=tree[k<<1];
    }else{
        tree[k]=tree[k<<1|1];
    }
```

```
}
Node query(int i,int j,int k){
    if(x<=i && j<=y){
        return tree[k];
    }
    int mid=i+j>>1;
    Node ret;
    if(x<=mid){
        Node tmp=query(i,mid,k<<1);
        if(ret.mmm<tmp.mmm) ret=tmp;
    }
    if(y>mid){
        Node tmp=query(mid+1,j,k<<1|1);
        if(ret.mmm<tmp.mmm) ret=tmp;
    }
    return ret;
}

int a[MAXN];
long long d[MAXN];
int main(){
    int n;
    scanf("%d",&n);
    for(N=1; N<n; N<<=1);
    for(int i=1; i<n; ++i){
        scanf("%d",a+i);
        x=i; y=a[i];
        update(1,N,1);
    }
    x=n; y=n;
    update(1,N,1);
    for(int i=n-1; i>=1; --i){
        x=i+1; y=a[i];
        Node tmp=query(1,N,1);
        d[i]=d[tmp.idx]+n-i-(a[i]-tmp.idx);
    }
    long long res=0;
    for(int i=1; i<=n; ++i){
        res+=d[i];
    }
    printf("%lld",res);
    return 0;
}
```

Bonus problem :

12-Lazy Propogation :

```
#include<bits/stdc++.h>
using namespace std;

void buildTree(int* arr,int* tree,int start,int end,int treeNode){

	if(start == end){
		tree[treeNode] = arr[start];
		return;
	}
	int mid = (start+end)/2;

	buildTree(arr,tree,start,mid,2*treeNode);
	buildTree(arr,tree,mid+1,end,2*treeNode+1);
	tree[treeNode] = min(tree[2*treeNode],tree[2*treeNode+1]);
}

void updateSegmentTreeLazy(int* tree,int* lazy,int low,int high,int startR,int endR,int
currPos,int inc){

	if(low > high){
		return;
	}

	if(lazy[currPos] !=0){
		tree[currPos] += lazy[currPos];

		if(low!=high){
			lazy[2*currPos] += lazy[currPos];
			lazy[2*currPos+1] += lazy[currPos];
		}
		lazy[currPos] = 0;
	}

	// No overlap
	if(startR > high || endR < low){
		return;
	}

	// Complete Overlap

	if(startR<= low && high <= endR){
		tree[currPos] += inc;
```

```
42.                     if(low!=high){
43.                             lazy[2*currPos] += inc;
44.                             lazy[2*currPos+1] += inc;
45.                     }
46.                     return;
47.             }
48.
49.             // Partial Overlap
50.
51.             int mid = (low+high)/2;
52.             updateSegmentTreeLazy(tree,lazy,low,mid,startR,endR,2*currPos,inc);
53.             updateSegmentTreeLazy(tree,lazy,mid+1,high,startR,endR,2*currPos+1,inc);
54.             tree[currPos] = min(tree[2*currPos],tree[2*currPos+1]);
55. }
56.
57. int main(){
58.
59.             int arr[] = {1,3,-2,4};
60.             int* tree = new int[12]();
61.             buildTree(arr,tree,0,3,1);
62.             int* lazy = new int[12]();
63.             updateSegmentTreeLazy(tree,lazy,0,3,0,3,1,3);
64.             updateSegmentTreeLazy(tree,lazy,0,3,0,1,1,2);
65.
66.             cout<< "Segment Tree" <<endl;
67.             for(int i=1;i<12;i++){
68.                     cout<<tree[i]<< endl;
69.             }
70.
71.             cout<< "Lazy Tree" <<endl;
72.             for(int i=1;i<12;i++){
73.                     cout<<lazy[i]<< endl;
74.             }
75.
76.             return 0;
77. }
```