STRBIT - Editorial

6

PROBLEM LINK:

Practice
Contest

Author: Vitaliy Herasymiv
Tester: Istvan Nagy
Editorialist: Amit Pandey

DIFFICULTY:

Easy.

PREREQUISITES:

BIT, segment tree.

PROBLEM:

Determine minimum amount of minutes required to paint a fence consisting of N parts, in green color. Each part is initially painted either red or green. We can chose an index X, and flip colors in range $[X, \min(X+K, N-1)]$ in one minute.

QUICK EXPLANATION:

Consider the index where first red is occurring as T, flip the colors in range $[T, \min(N, T+K-1)]$. Keep repeating this step until each part is colored green.

EXPLANATION:

First lets see how to solve the problem, later we can see the proof of correctness.

Consider the index at which first red is occurring and call it T, flip the colors in range $[T + \min(N, T+K-1)]$. Keep repeating the given procedure unless whole array is green. After each iteration value of T will be increasing, so the given algorithm will terminate in less than or equal to N steps.

Each iteration may take $O(N)$ amount of time, hence the given algorithm will be taking $O(N^2)$ time. As the N can be 10^5 in the given problem, we need to optimize our solution.

We can solve the problem in $O(N \log N)$ time. Range flip can be seen as adding 1 to the range and value of color can be obtain using sum at that particular index modulo 2. If the $\text{sum}\%2$ is 0, then color at that index will be same as the initial one or flipped otherwise. The range updates and queries can be done in $O(\log N)$ time using BIT or segment tree.

O(N) Solution:

It can be done in $O(N)$ as well. Adding 1 to range $[L, R]$ can be seen as adding 1 to index L and adding -1 to index $(R+1)$. When we want to retrieve the value at specific index, we need to take the prefix sum upto that index. See sub problem 1 of this editorial for more details.

Proof:

The given procedure can be proved correct using the following lemmas.

1. Order of the updates doesn't matter.
2. Starting index of any 2 updates will be different, as 2 updates at same starting position will cancel each other.
3. If we sort the minimum update sequence and suppose first update happens at index i, then at each index in range $[0, i-1]$ color must be green and at index i, color must be red.

Solution:

Setter's solution can be found here
Tester's solution can be found here

segment-tree strbit bit cook56 easy

This question is marked "community wiki".

edited 11 Feb, 18:39
admin ♦♦
[12.1k]●346●481●496

asked 21 Mar '15, 02:10
amitpandeykgp
[259]●11●22
accept rate: 0%

Tags:

easy ×1,642
segment-tree ×632
bit ×241
cook56 ×122
strbit ×1

Asked: 21 Mar '15, 02:10
Seen: 3,303 times
Last updated: 11 Feb, 18:39

Related questions

SUBARR - Editorial
MQRY Editorial
How to augment a rooted tree using a segment tree or Binary Indexed Tree(with regards to QTREE6...
CHEFAXR - Editorial
SPREAD - Editorial
TAEDITOR - Editorial
Need resources for Binary index tree, Segment tree
A Technique to convert Tree to a Linear Array for efficient query processing .
FBCHEF - Editorial
VISITALL - Editorial

8 Answers: oldest newest most voted

7

It can also be solved with a queue in $O(N)$.

Scan left to right, and in the queue, keep the locations of all flips in the last K parts. When processing a part i, first pop from the front of the queue while the element at the front of the queue x satisfies $|x - i| \geq K$. If a part is green and the queue's size is odd, or if a part is red and the queue's size is even, add i to the queue, and add 1 to the answer.

link | award points

answered **23 Mar '15, 00:10**
waterfalls
[141]●2
accept rate: 66%

2

I solved it without using BIT or segment tree.
We can solve this problem using this **O(N)** algorithm:

1. Initialise st[] = -1, st[i] denotes total number of swaps when **i** index was part of a range 1st time.
2. Iterate over the char array, s[]:
a. If st[i]=-1 then there is no swaps till the current index.
b. Else calculate the number of swaps on current index as (ans-st[i]), here ans=number of swaps till index **i**.
c. Calculate value of s[i] after catering the swaps.
d. If value of s[i] after catering swaps is 'R' then assign the new elements in range with the current value of ans. Increase the value of ans if s[i]='R'.

Code:

```
        memset(st,-1,sizeof(st));
    for(i=0; i < n ; i++)
        {
            if(st[i]!=-1)
            {
                j=ans-st[i];
                if(j&1 && s[i]=='R')
                    x='G';
                else if(j&1 && s[i]=='G')
                    x='R';
                else
                    x=s[i];
            }
            else
                x=s[i];
            if(x=='R')
            {
                j=min(n-1,i+k-1);
                while(st[j]==-1 && j>=i)
                {
                    st[j]=ans;
                    j--;
                }
                ans++;
            }
        }
        pi(ans);
```

link | award points

edited **23 Mar '15, 00:30**

answered **23 Mar '15, 00:24**

atulsehgal
[451]●6●12
accept rate: 8%

1

My solution is a bit simpler:

```
int n, k;
char s[100010];
int flip_bound[100010];

int main() {
  int t;
  cin >> t;
  while (t--) {
    cin >> n >> k;
    cin >> s;
    int ans = 0;
    CLR(flip_bound, 0);
    int cur_flip = 0;
    REP(i, n) {
      if ((s[i] == 'R') == (cur_flip % 2 == 0)) {
        ans++;
        cur_flip++;
```

```
        if (i + k <= n) {
          flip_bound[i + k - 1] = 1;
        }
      }
      if (flip_bound[i]) {
        cur_flip--;
      }
    }
    cout << ans << endl;
  }
  return 0;
}
```

link | award points

answered **23 Mar '15, 00:41**
lxfind
[291]●1●4●7
accept rate: 14%

1

There is no need of segment tree or bit .

http://www.codechef.com/viewsolution/6566527

This problem can be solved by looping from 0 to n-1 and increment the answer whenever current alphabet is 'R' and number of flips are even or current letter is 'B' and number of flips are odd . Complexity O(n) .

Pseudo code :

Input : arr , n , k // string , string size , sub string size which can be flipped

answer := 0 , flips :=0 , mark[]:={0}

for i from 0 to n-1 :

flips = flips + mark[i]

if arr[i] is 'R' and flips is even OR arr[i] is 'G' and flips is odd :

increment answer by 1

mark[i+1] = mark[i+1] + 1

mark[i+k] = mark[i+k] - 1 // marking the subsegment from i to i+k-1 to be flipped

end if

end for

print answer

link | award points

answered **23 Mar '15, 17:57**

rajat1603
[528]●10
accept rate: 0%

For those who don't understand what he did , I am giving some clues.

Actually he maintained a mark array corresponding to the input string.When we encounter 'R' , we check if corresponding flips at that position is even. If it's even we have to flip it so we add +1 to our answer. Similarly for Green we add +1 to answer if corresponding number of flips are odd.

Now , we have to maintain the array if we do the flip and add +1 to our resulting minutes. For this , we are marking the array at points (i , i+k-1).

shubham99 (24 Dec '15, 04:27)

0

@rajat1603, This problem can be solved by looping from 0 to n-1 and increment the answer whenever current alphabet is 'R' and number of flips are even or current letter is 'B' and number of flips are odd . Complexity O(n) .

Hey,I am not able to feel your conclusion. Please help...

link | award points

answered **24 Mar '15, 16:40**
khooni_khopri
[127]●2●13
accept rate: 0%

0

@rajat1603 can you please explain the idea behind your algo??

EDIT: Got it. Awesome solution

link | award points

edited **28 Mar '15, 23:15**

answered **28 Mar '15, 22:46**
esemzv
[21]●2
accept rate: 0%

@amitpandeykgp I am not getting the Sub-problem 1 of the editorial you mentioned here.

0

Let the array be: a[]= {1,2,4,1,2,1}, X=3 and b[] be the updated array a[] at each step

update[1,3]=> a[]= {4,2,4,-2,2,1} //a[1]+=3 n a[4]-=3 while b[]= {4,5,7,1,2,1}

update[2,4]=> a[]= {4,5,4,-2,-1,1} //a[2]+=3 n a[5]-=3 while b[]= {4,8,10,4,2,1}

update[3,6]=> a[]= {4,5,7,-2,-1,1} //a[3]+=3 n a[7]-=3 while b[]= {4,8,13,7,5,4}

```
     prefix_sum[]= {4,9,16,14,13,14}  //i don't see how prefix array contains correct no. at each
index
            sum[]= {4,13,29,43,56,70}
```

query[1,4]= sum[4]-sum[0]=43 while from b[1,4] we get 32

query[2,5]= sum[5]-sum[1]=52 while from b[2,5] we get 33

I have read it 2-3 times and still not getting. Please help.

link | award points edited **30 Mar '15, 01:50** answered **30 Mar '15, 01:48**

moulin
[1]•1
accept rate: 0%

---

can some one explain last comment please how we get query result correctly

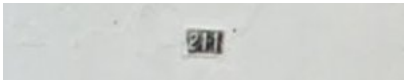
0

link | award points answered **28 May '15, 00:03**

raj__0jaiswal
[1]
accept rate: 0%

Your answer

[hide preview] community wiki



Type the text

Privacy & Terms

Post Your Answer