July 12, 2025

STL APPLICATION IDEA

Ques 1. https://atcoder.jp/contests/abc237/tasks/abc237_d

Problem Statement

There is a sequence that contains one 0, $A = (0)$.

Additionally, you are given a string of length N, $S = s_1 s_2 \ldots s_N$, consisting of L and R.

For each $i = 1, 2, \ldots, N$ in this order, the following will be done.

- If s_i is L, insert i to the immediate left of $i - 1$ in A.
- If s_i is R, insert i to the immediate right of $i - 1$ in A.

Find the final contents of A.

Sample Input 1 Copy

```
5
LRRLR
```

Sample Output 1 Copy

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

Initially, $A = (0)$.
S_1 is L, which makes it $A = (1, 0)$.
S_2 is R, which makes it $A = (1, 2, 0)$.
S_3 is R, which makes it $A = (1, 2, 3, 0)$.
S_4 is L, which makes it $A = (1, 2, 4, 3, 0)$.
S_5 is R, which makes it $A = (1, 2, 4, 5, 3, 0)$.

→ Insertion is happening around current last element.

Ex: LRRLR

(i) L

We have 0 initially, so where will 1 go? To the left of it

last el

0

1

incoming

1, 0

(ii) R

last element inserted = 1.

So 2 will go right to it

1 →R 2, 0

AlgoZenith

iii) R

last element inserted 2.

3 will go right to 2

1, 2, 3, 0

iv) L

last element inserted = 3

4 will be inserted to the left to 3

1, 2, 4, 3, 0

v) R

Last element inserted = 4

5 will go towards right of 4.

⇒ 1, 2, 5, 4, 3, 0

Approach :- Create 2 vectors, one for right & one for left. Initially its empty.

- Go through characters of s one by one.
- For each character, if new value has to go to right then current element will be pushed to left and vice versa for when new value has to go left.
- Simply print the 2 vectors. We will print right in reverse.

```cpp
vector<int> left,right;
int cur = 0;
for(auto v:s){
    if(v=='L'){
        right.push_back(cur);
    }else{
        left.push_back(cur);
    }
    cur++;
}

for(auto v:left){
    cout<<v<<" ";
}
cout<<cur;
reverse(right.begin(),right.end());
for(auto v:right){
    cout<<" "<<v;
}
```

Ques 2. https://atcoder.jp/contests/abc158/tasks/abc158_d

Problem Statement

Takahashi has a string S consisting of lowercase English letters.

Starting with this string, he will produce a new one in the procedure given as follows.

The procedure consists of Q operations. In Operation i $(1 \leq i \leq Q)$, an integer T_i is provided, which means the following:

- If $T_i = 1$: reverse the string S.
- If $T_i = 2$: An integer F_i and a lowercase English letter C_i are additionally provided.
  - If $F_i = 1$: Add C_i to the beginning of the string S.
  - If $F_i = 2$: Add C_i to the end of the string S.

Help Takahashi by finding the final string that results from the procedure.

Deque will be used as we have to insert & remove from back & front.

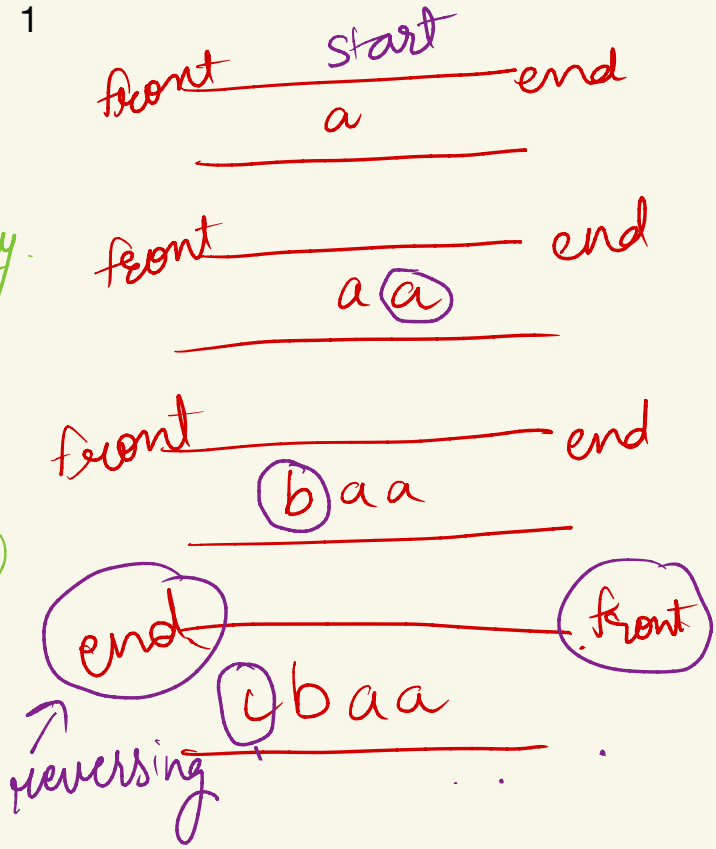
In reversing, we could relabel front as back and back as front.

Ex:
```
a   <- starting string
6
2 2 a
2 1 b
1
2 2 c
1
1
```

```cpp
int flipped = 0; // Maintaining if its rotated or not so we can insert accordingly.

for(int i=0;i<q;i++){
    int t;
    cin>>t;
    if(t==1){
        // reverse
        flipped = (1+0)-flipped; //O(1)
    }else{
        int f;char c;
        cin>>f>>c;
        if(flipped){
            f = (1+2)-f;
        }
        if(f==1)dq.push_front(c);
        else dq.push_back(c);
    }
}
if(flipped)reverse(dq.begin(),dq.end());
for(auto v:dq){
    cout<<v<<" ";
}
```

O(1)



⇒ aabc

Outputting the dq in reverse

Ques 3> https://codeforces.com/problemset/problem/821/C

Okabe and Super Hacker Daru are stacking and removing boxes. There are n boxes numbered from 1 to n. Initially there are no boxes on the stack.

Okabe, being a control freak, gives Daru $2n$ commands: n of which are to add a box to the top of the stack, and n of which are to remove a box from the top of the stack and throw it in the trash. Okabe wants Daru to throw away the boxes in the order from 1 to n. Of course, this means that it might be impossible for Daru to perform some of Okabe's *remove* commands, because the required box is not on the top of the stack.

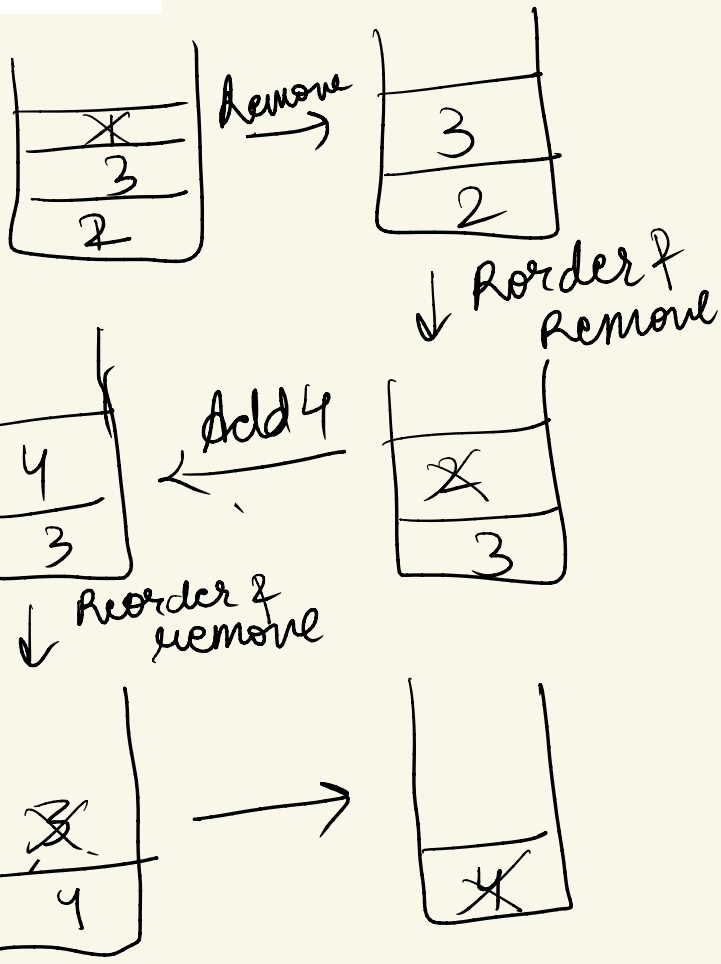
That's why Daru can decide to wait until Okabe looks away and then reorder the boxes in the stack in any way he wants. He can do it at any point of time between Okabe's commands, but he can't add or remove boxes while he does it.

Tell Daru the minimum number of times he needs to reorder the boxes so that he can successfully complete all of Okabe's commands. It is guaranteed that every box is added before it is required to be removed.

Have to do this in O(N)

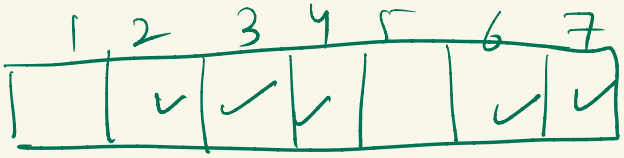
Add 2
A 3
A 1

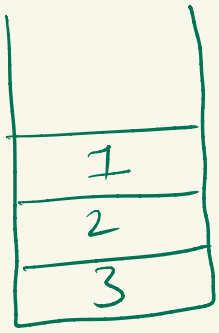
Remove (R) → 1
R → 2 (this needs to be rem) but its not at top so reorder
4
R → 3
R → 4



Approach:

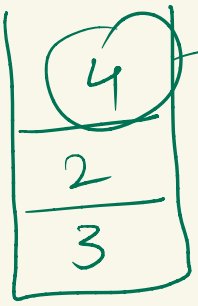
If you reordered once, we don't have to explicitly reorder it again unless we get a new number in stack. which will disrupt the order. So, when conflict happens once, we remove all elements from stack since they are at the right place.



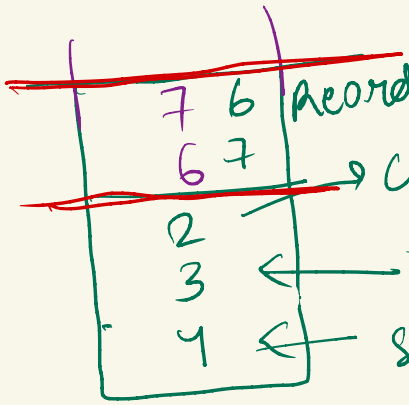


cur = 1

R, Add 4



creates conflict



reorderd & sorted

can we remove this as it has been reordered

this can also be removed similarly

same with this.

When we are reordering everytime $\Rightarrow O(n^2)$

Clear the stack and remove only if stack is non empty

$\Rightarrow O(n)$

Ques 4. https://atcoder.jp/contests/abc206/tasks/abc206_c

Given an array of N integers $A = (A_1, A_2, \ldots, A_N)$, find the number of pairs (i, j) of integers satisfying all of the following conditions:

- $1 \leq i < j \leq N$
- $A_i \neq A_j$

for every j, we are finding the feasible i

↓

done using map

```
int main(){
  int n;
  cin>>n;
  int arr[n];
  for(int i=0;i<n;i++){
    cin>>arr[i];
  }
  long long ans = 0;
  map<int,int> freq;
  for(int j=0;j<n;j++){
    ans += j-freq[arr[j]];
    freq[arr[j]]++;
  }
  cout<<ans<<endl;

}
```

Homework Problems :

1. https://atcoder.jp/contests/abc170/tasks/abc170_e

2. https://atcoder.jp/contests/abc045/tasks/arc061_b