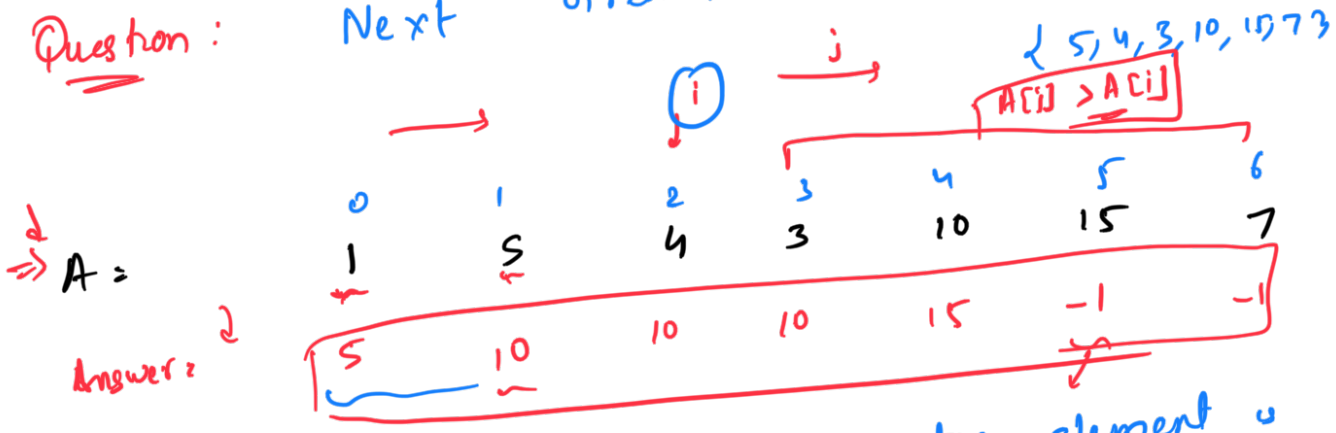
Stacks - 2

Question: Next Greater Element



=> For A[i], the next greater element A[j] such that $j > i$ && $A[j] > A[i]$

least j

(i)

A[i] would be the an

Brute Force:

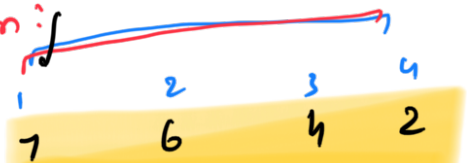
```
for(i=0; i<n; i++) {        flag = 0;
    ans[i] = -1;
    for(j=i+1; j<n; j++) {
        if( A[j] > A[i] ) {
            ans[i] = A[j];        // ans[i] = j
            flag = 1;
            break
        }
    }
    if(flag == 0) ans[i] = -1
}
```

T.C: $O(n^2)$

S.C: $O(1)$

Better Approach:

Observation:

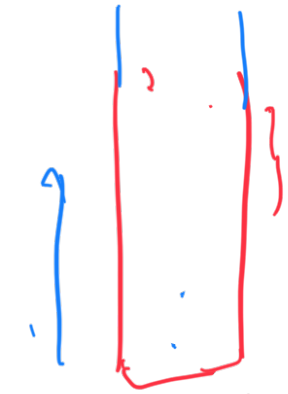


A =

n G =

A =

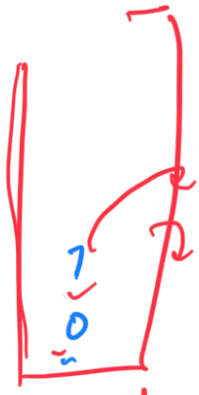
stack which has elements in decreasing order.



push: 1
pop : 1
2 operation on stack

A =

Ans =



stack.

A[3] > stk.top() ans[2] = 4

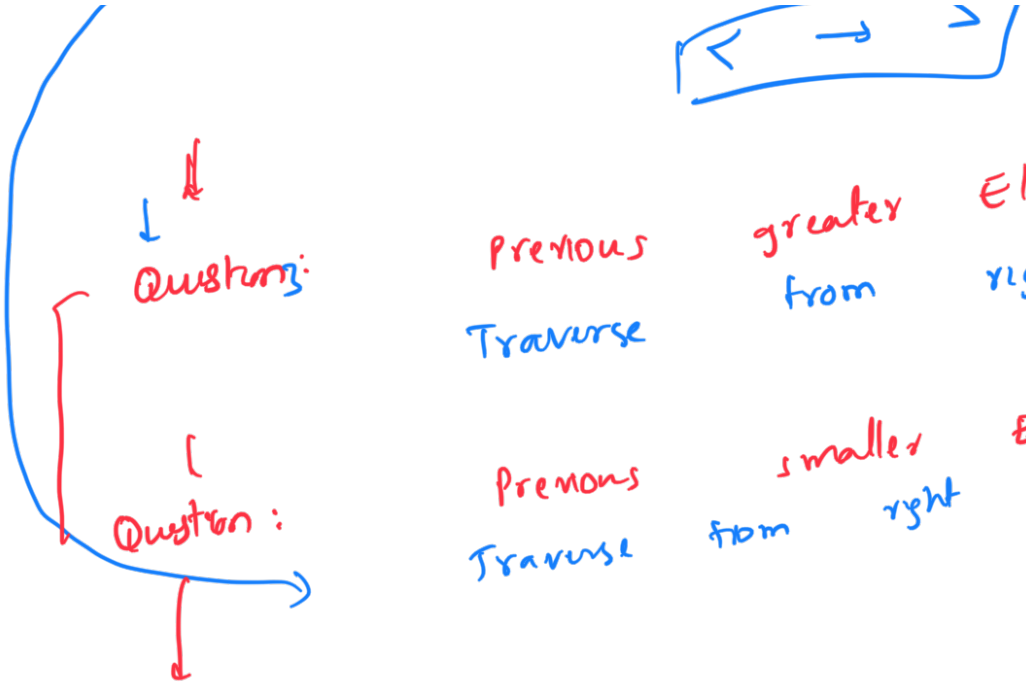
ind => ans[ind]

4 < Arr[top]

top = stk.pop();
ans[top] = A[i]

Question: Next Smaller Element.

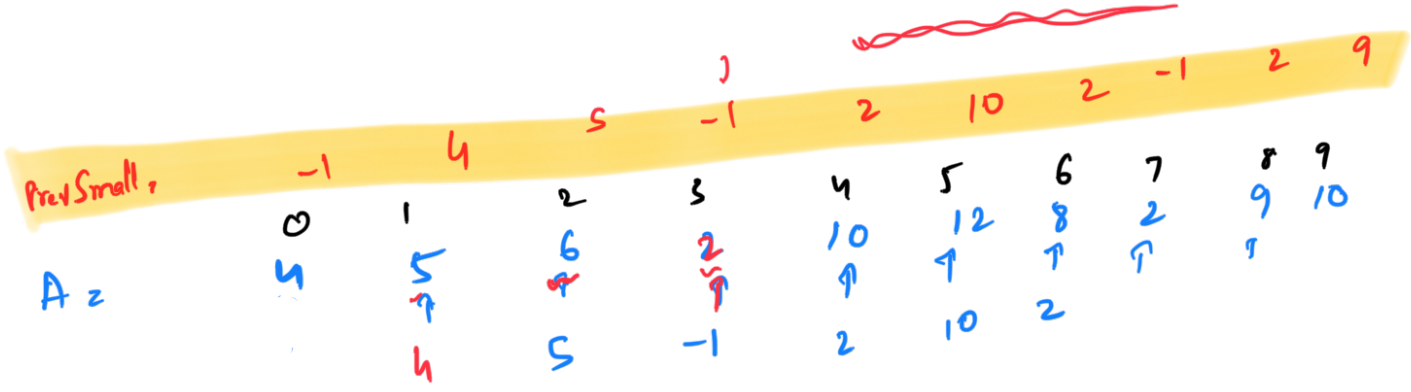
→ Stack will be in decreasing order

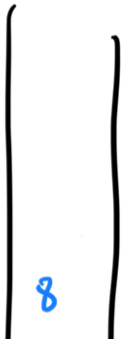


A = 9 2 6 8

= Prev Smalls = -1 -1 2 6

Approach 2: Previous Smaller / Greater Element





previous Previous No

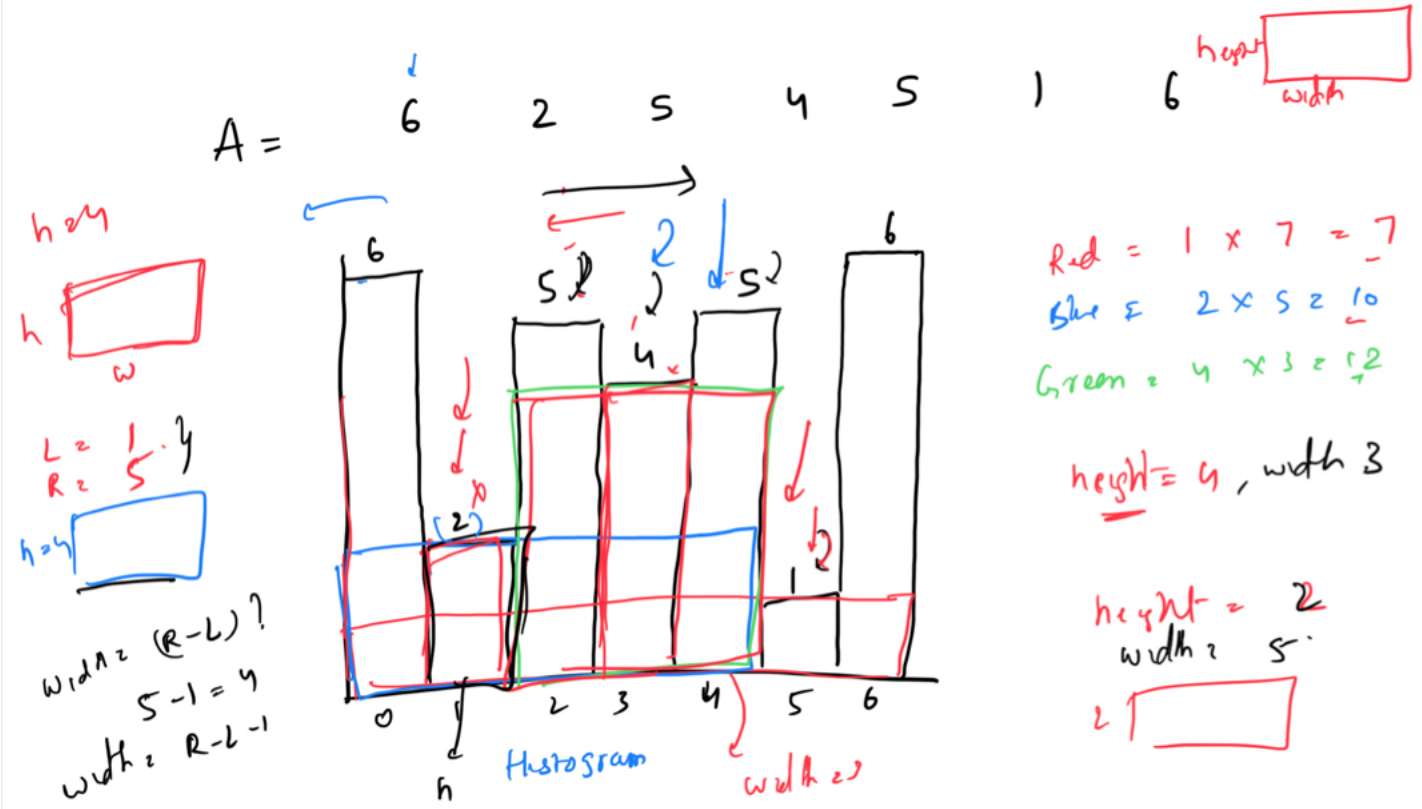
Finding the answer for A[i] by looking the Stack

T.C: O(n)

S.C: O(n)

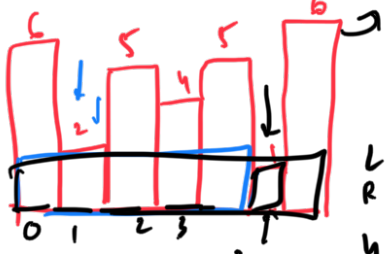
Question: Largest Rectangle in a Histogram

A = 6 2 5 4 5 1 6



Brute force

-> Let us consider every bar to be the height of rectangle



area = height x width.

L = -1
R = 5
width = R-L-1
= 5-(-1)-1
= 5

L = -1
R = 7

L = -1, R = 7
R-L-1 = 7-(-1)-1
= 7

	0	1	2	3	4	5	6
A :	6	2	5	4	5	1	6
	-1	-1	1	1	3	-1	5

prevSmaller =
nextSmaller =
stacks

	0	1	2	3	4	5	6
	-1	-1	1	1	3	-1	5
	1	5	3	5	5	7	7

prevSmaller = {-1}, nextSmaller = {N}
ans = INT_MIN;

L = 5
R = 7
R-L-1 = 7-5-1
= 1

```
for(i=0; i<n; i++) {
    // A[i] is the height of rectangle.
    R = nextSmaller[i];
    L = prevSmaller[i];
    width = R-L-1;
    ans = max(ans, A[i] x width);
                    (area)
}
```

O(n)

T.C: $O(n^2)$

S.C: $O(n)$

Question: Max and Min

Find the summation of difference of max and min of all the subarrays

Subarrays = $\frac{n(n+1)}{2}$

[3]

A = [2 5 3]

$\frac{3 \times 4}{2} = 6$

Subarrays	Max	Min	Difference
[2]	2	2	0
[5]	5	5	0
[3]	3	3	0
[2, 5]	5	2	3
[5, 3]	5	3	2
[2, 5, 3]	5	2	3

3 + 2 + 3 = 8

Brute Force:

Consider all subarrays : $\frac{n(n+1)}{2}$

T.C for 1 subarray $O(n)$

Total T.C: $O(n^2 \times n) = O(n^3)$

S.C: $O(1)$

Ans = (2)(2) + (5)(5) + (3)(3) + (5)(2) + (5)(3) + (5)(2)

ele	max times	min times
2	1	3
5	4	1
3	1	2

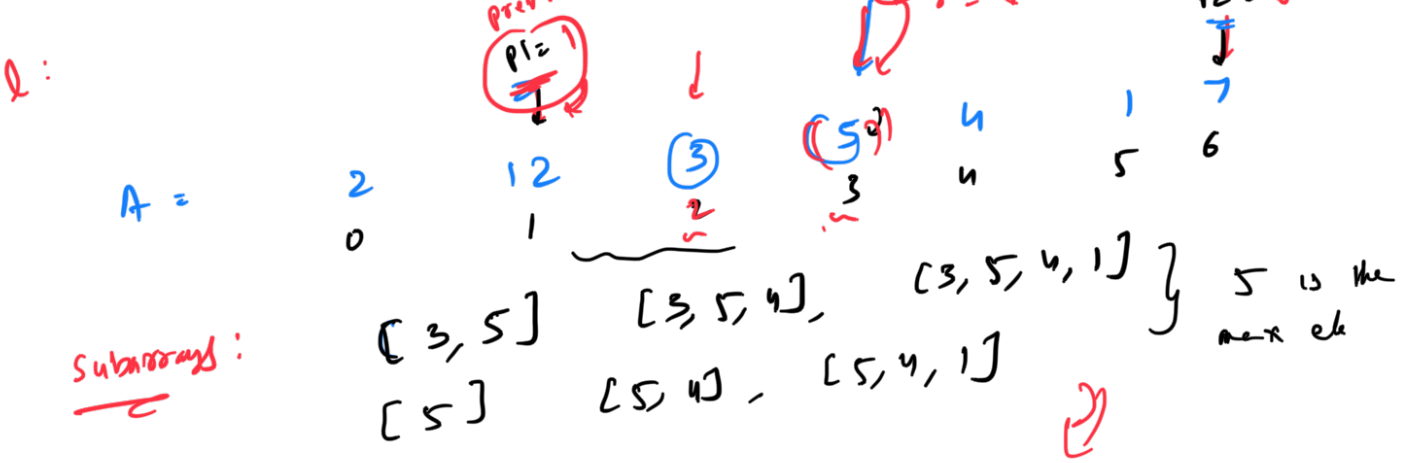
Ans = 2*1 − 2*3 + 5*4 − 5*1 + 3*1 − 3*2

(A[i] × maxTimes − A[i] × minTimes)

Ans = Sum

→ How to find no. of subarrays in which A[i] is max / min?

l:



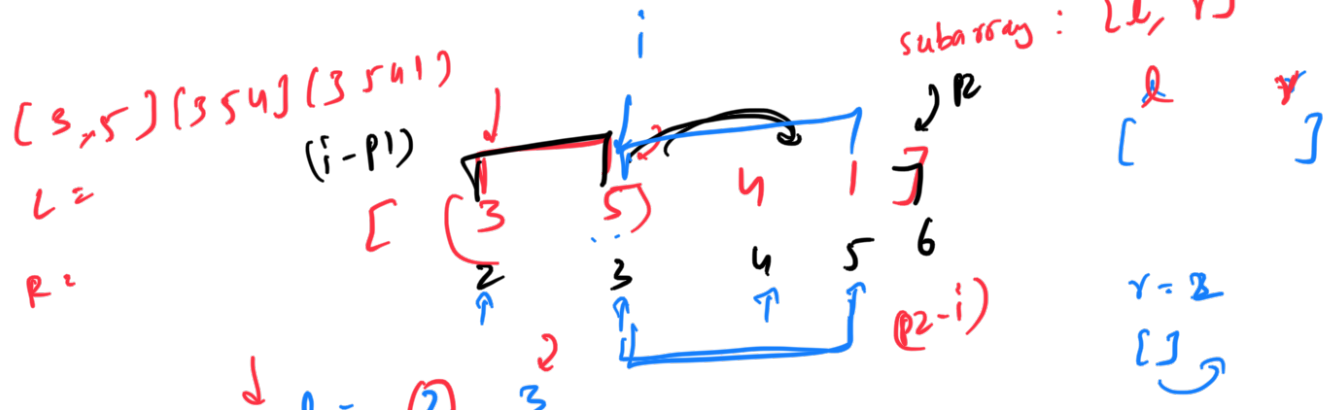
subarrays:

[3, 5] [3, 5, 4] [3, 5, 4, 1] } 5 is the max el
[5] [5, 4] [5, 4, 1]

6 subarrays

[3, 5, 4, 1] ⇒ $\frac{n(n+1)}{2} = \frac{4 \times 5}{2} = 10$

→ Count No. of subarrays using elements [3, 5, 4, 1] which definitely has 5 in it.



l = 2, 3

r = 5, 4, 3

2 × 3

total subarrays =

for any element i,

prev greater elemet.

p1 = Index of next smaller elem

p2 = Index of next greater elem

times = $(i-p1) \times (p2-i)$

[3, 5, 4, 1] =

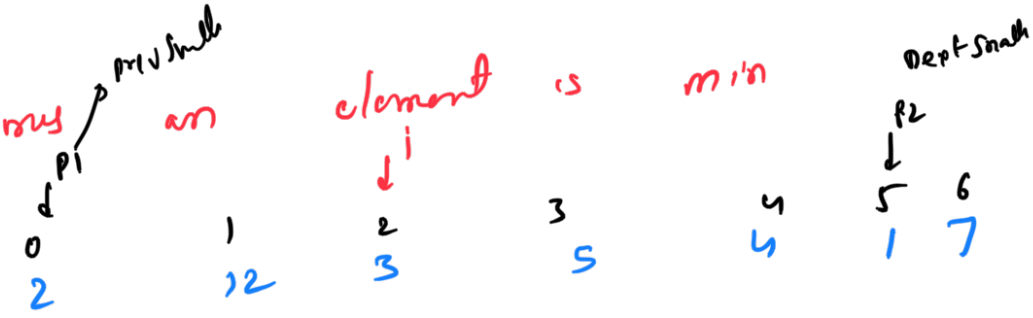
[3] [4,1]

$\frac{4 \times 5}{2} = 10 - \frac{1(1+1)}{2} - \frac{2 \times 3}{2}$

$= 10 - 1 - 3$

$= 10 - 4 = 6$

No. of times an element is min



A =

[12,3] [12,3,5] [12,3,5,4] } = 8

[3] [3,5] [3,5,4]

Start Points = $(i-p1)$

end Points = $(p2-i)$

Subarray = $(i-p1) \times (p2-i)$

prevGreater, prevSmaller = [-1]*n

nextGreater, nextGreater = [n]*n

for (i = 0; i < n; i++) {

p1 = prevGreater[i];

p2 = nextGreater[i];

sum += $(i-p1)(p2-i) \times A[i]$

p1 = prevSmaller[i];

p2 = nextSmaller[i] =

$(i-p1)(p2-i) \times A[i]$;

O(n)

sum = =

}

S.C: O(n)

T.C: O(n)

Question: Arithmetic Expression Evaluation

Expression: combination of operands & operator

5 + 1, p + q, a + 5

1) Infix
2) Prefix (Polish)
3) Postfix (Notation)

Infix:

<operand1> operator <operand2>

→ a - b

(x-y) - (p+q)

BODMAS

((5 + 1) * 6) = ?

5 + 6 = 11

36

(5+1) * 6 = 36

Precedence

1) ( ) [ ] { }
2) ^ → Power Operation
3) * /
n) + -

$a + b * (c \hat{} d - e) \hat{} (f + g * h) - i$

⇒

step1: $a + b * k_1 \hat{} k_2 - i$

step2: $a + b * k_3 - i$

step3: $(a + k_4) - i$

step4: $k_5 - i$

step5: k_6

} Multiple steps

2) Prefix (Polish Notation)

<operator> <operand1> <operand2>

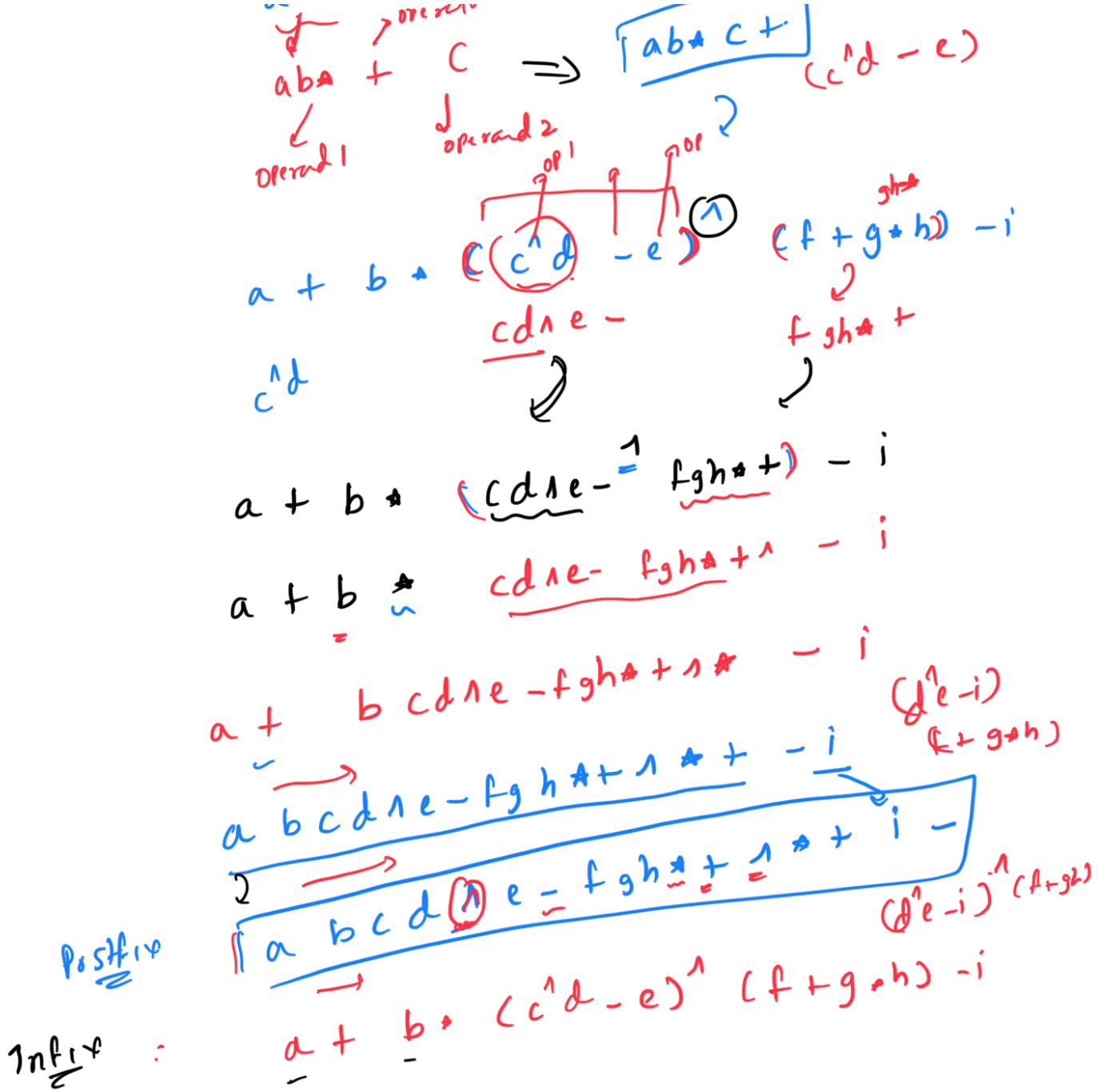
$a + b \Rightarrow +ab$

3) Postfix (Reverse Polish)

<operand1> <operand2> <operator>

$a - b \Rightarrow ab -$

$a * b + c \Rightarrow$



Observations

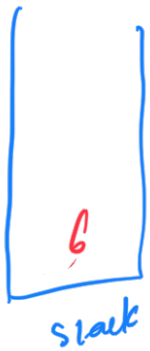
1) All operands come in same order

2) We do operations in the same order as in postfix

→ Computer stores in prefix / postfix form so that it can evaluate expression in just 1 iteration

Question: Evaluate Postfix Expressions

postfix: 4 6 + 3 1 - 2 * -



10-4=6

ele1 = 6
ele2 = 4

6+4 = 10

2*2=4

3-1 = 2.

(A)
(a^b)

T.C: O(n)
S.C: O(n)

Question: Double character Trouble

S = a b c d e e d c ~~a a~~ b ~~x x~~ d a d

Brute force

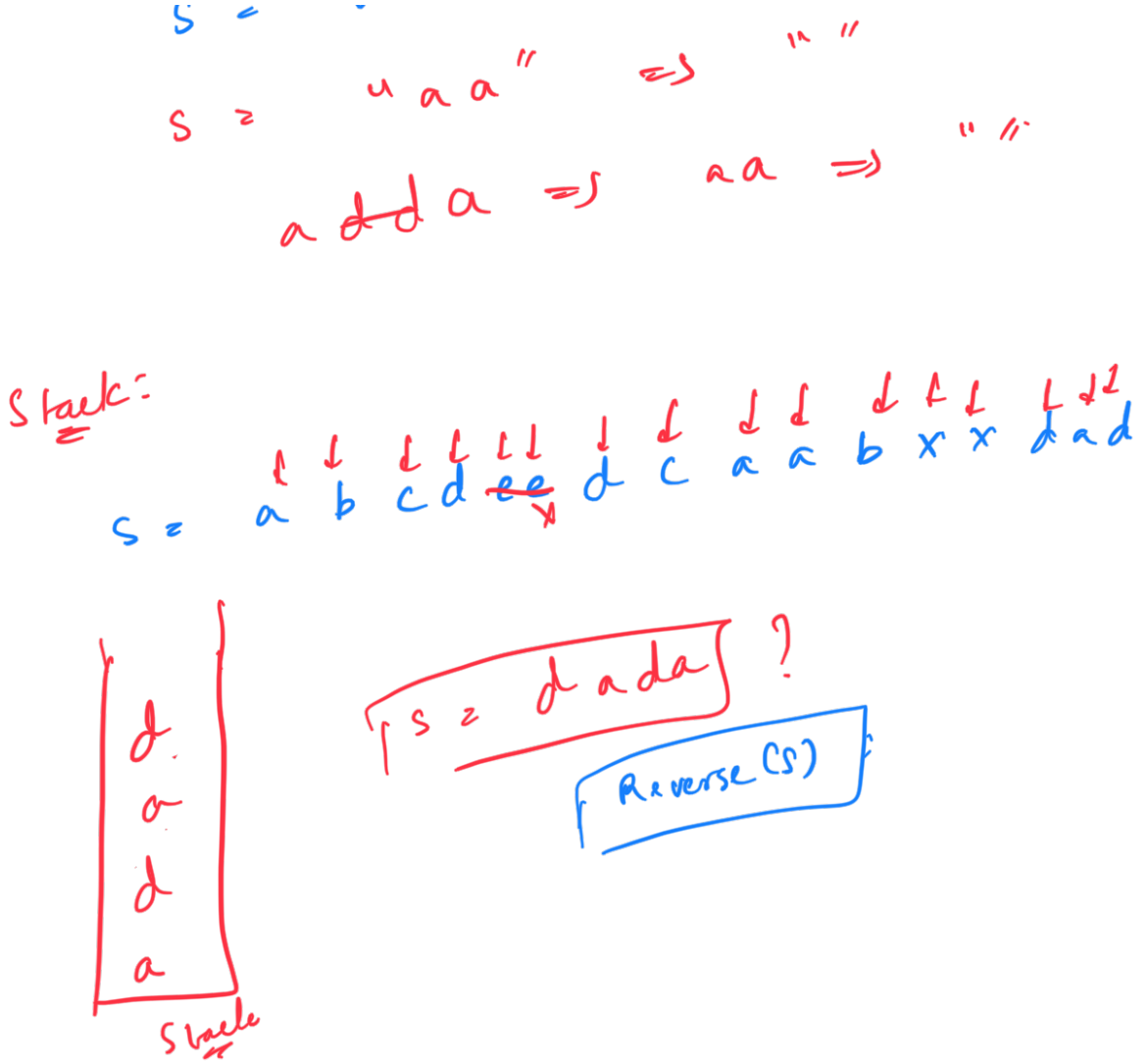
S = a b c ~~d d~~ c b d a d

S = a b ~~e e~~ b d a d

S = a ~~b b~~ d a d

S = a d a d

~~a a~~ a" -> "a"



Question: Double Character Trouble

```
string doubleCharacterTrouble(string s){
      stack<char> stk;
      for(int i = 0; i < s.size(); i ++){
            if(stk.isEmpty() || stk.top() != s[i])
                  stk.push(s[i]);
            else
                  stk.pop();
      }
      return reverse(stk);
}
```

Question: Postfix expression evaluation

```
int evalExpr(string s){
      stack<char> stk;
      for(int i=0; i < s.size(); i++ ){
            if(s[i] is a character number)
                  stk.push(s[i]);
            else{
                  op1 = stk.top();
                  stk.pop();
                  op2 = stk.top();
                  stk.pop();
                  stk.push(int(op1) <operator> int(op2));
            }
      }
      return stk.top();
}
```

Question: Next greater element

```
void nextGreater(int a[], int n){
      stack<int> stk;
      int ans[];
      for(int i = 0; i < n; i++){
            if(stk.isEmpty() || a[i] <= a[stk.top()])
                  stk.push(i);
            else{
                  while(!stk.isEmpty() && a[i] > a[stk.top()]){
                        ans[stk.top()] = a[i];
                        stk.pop();
                  }
                  stk.push(i);
            }
      }
      // Remaining elements do not have a greater element
      while(!stk.isEmpty()){
            ans[stk.top()] = -1;
            stk.pop();
      }
      return;
}
```

```
void previousSmaller(int a[], int n){
      stack<int> stk;
      int ans[];
      for(int i = 0 ; i < n ; i ++){
            while(!stk.isEmpty() && a[stk.top()] >= a[i]){
                  stk.pop();
            }
            if(stk.isEmpty())
                  ans[i] = -1;
            else
                  ans[i] = a[stk.top()];
            stk.push(i);
      }
return;
}
```

Question: Largest rectangle

```
preSmaller[] = {-1}, nextSmaller[] = {n};
max_ans = INT_MIN;
int rectangleArea(int arr[], int n){
      for(int i = 0; i < n ; i ++){
            l = prevSmaller[i];
            r = nextSmaller[i];

            ans = A[i] * (r - l - 1)
            max_ans = max(max_ans, ans);
      }
      return max_ans;
}
```

Question: Max and Min

```
int maxMin(int a[], int n){
      sum = 0;
      for(int i = 0 ; i < n; i ++){
            p1 = prevGreater[i]; // Get the index
            p2 = nextGreater[i];
            num_arrays = (i - p1) * (p2 - i);
            sum += num_arrays * a[i];

            p1 = prevSmaller[i]; // Get the index
            p2 = nextSmaller[i];
            num_arrays = (i - p1) * (p2 - i);
            sum -= num_arrays * a[i];
      }
return sum;
}
```