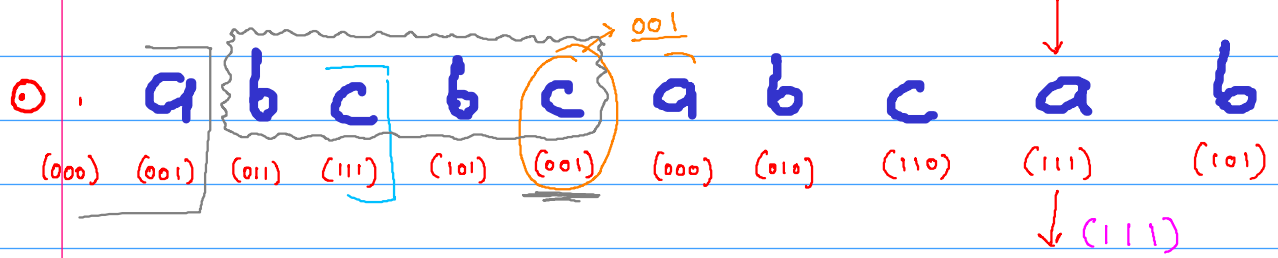
a b c



i → a b c
o/e o/e o/e

odd = 1
even = 0

C → even
b → even
a → odd

a even
b even
c even

(111)

a → odd
b → odd
c → odd

→ odd
→ odd
→ odd

odd - odd = even

2m+1 - (2m+1) = 2(n-m) even

odd - odd/even = even

odd - even

2m+1 - 2n = 2(m+n)+1

odd

```
INT  Hash = 0; ans=0;
MP[0] = 1;
for (auto c: S){
    Hash ^= (1<<(c-'a'));
    ans += MP[Hash];  MP[Hash]++;
}
return ans;
```

```
for( i  0-n)
    for( j  i-n)
    //for ( i-j ) even
        ans++;
```

Time O(n³)
Space O(1)

a b c b c d a d

```
for (i . 0-n)
    for (j - i-n)
        if (even) ans++
```

Time: O(n²)
Space O(1)

a b c d
i (o o e e)

0 ——— i

a → odd
b → odd
c → even
d → even

odd → 1
even → 0

even

0 ——— i

j1 j4 j5 j2 j3

5

a → odd
b → odd
c → even
d → even

a (0001) | b (0011) c (0111) b (0101) c (0001) d (0001)
(0000)
1 ^ b a

(i)

hash ^= (1 << (i'th))

a ^ 1

{ 0^1 = 1
  1^1 = 0 }

int

Signed

31

32 [z] ... d 'c' 'b' 'a'
... 3 2 1 0

1915. Number of Wonderful Substrings

Medium 616 44 Add to List Share

A **wonderful** string is a string where **at most one** letter appears an **odd** number of times.

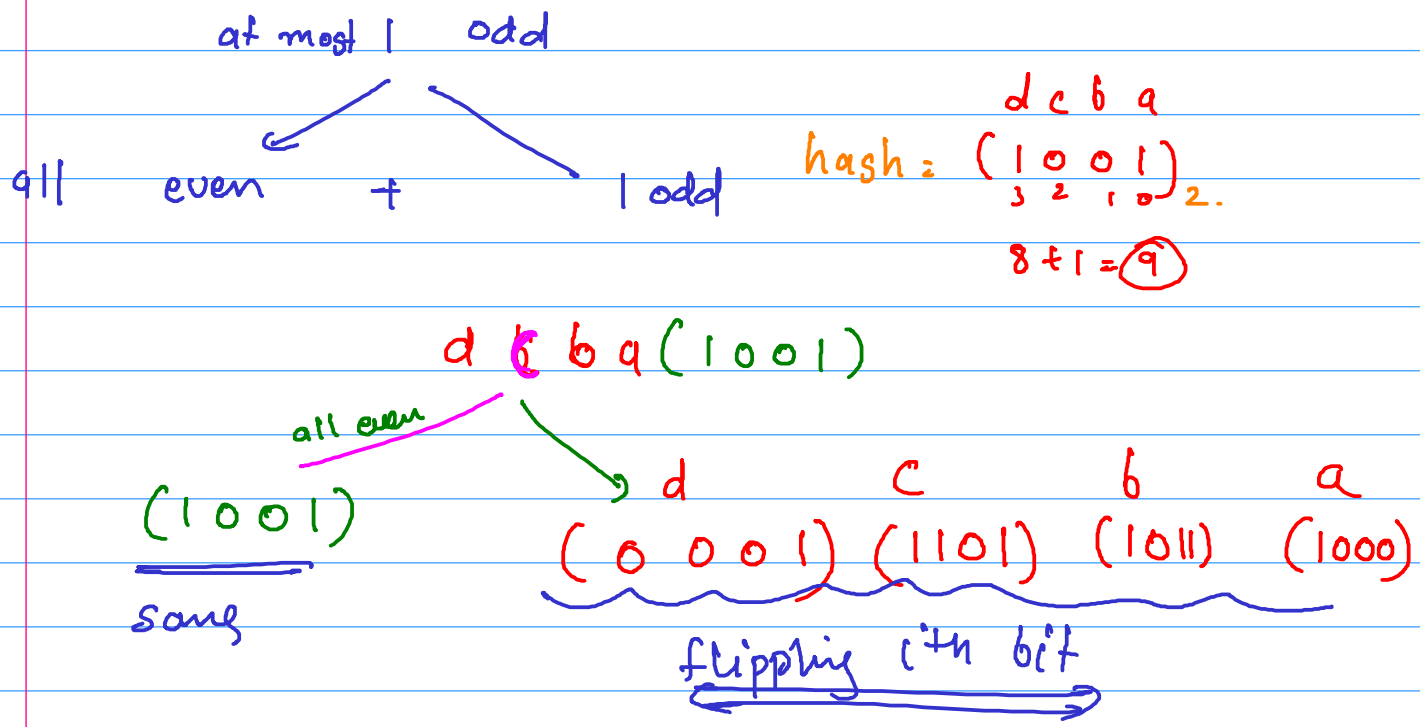
- For example, `"ccjjc"` and `"abab"` are wonderful, but `"ab"` is not.

Given a string `word` that consists of the first ten lowercase English letters (`'a'` through `'j'`), return *the **number of wonderful non-empty substrings** in `word`. If the same substring appears multiple times in `word`, then count **each occurrence** separately.*

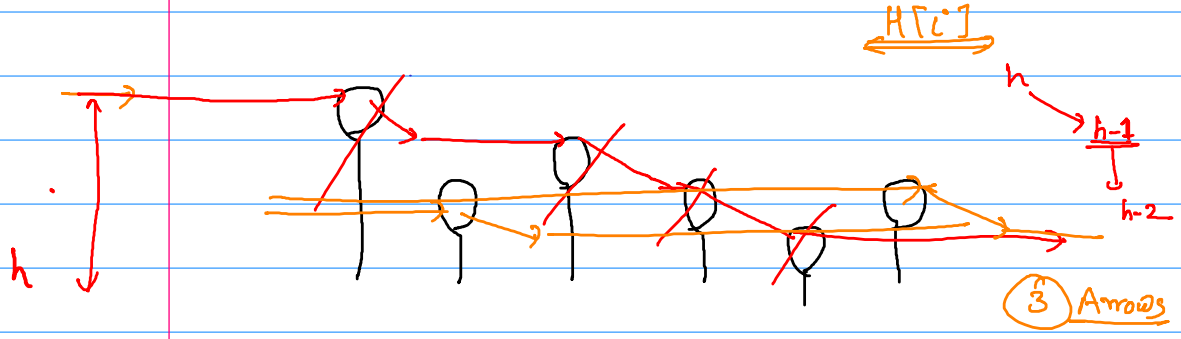
A **substring** is a contiguous sequence of characters in a string.

Example 1:

```
Input: word = "aba"
Output: 4
Explanation: The four wonderful substrings are underlined below:
- "aba" -> "a"
- "aba" -> "b"
- "aba" -> "a"
- "aba" -> "aba"
```



Q. [4 2 3 2 1 2]



[1, 2, 3, 4] = '4', [4, 3, 2, 1] = '1'

[4, [4], 3, [2], 1, 2]

int → set < >)
hf → set < > indices.
↳ search. removal
log N. log N

[4] → (1, 1)
[3] → (✗)
(2) → (✗, 5)
(1) → (✗)

[4] → ✗ 1
[3] → ✗ 0
[2] → ✗ 0
[1] → 1
[2] → 1

= 3 Answer

[4, [3], 2, 3, 2, 4]

ans = 1 + 1 + 1

ans: 3

[4] → ✗ 2 1
[3] → ✗ ✗ ✗ 0
[2] → ✗ 2

Q. Given a binary array. find maximum length subarray containing equal no. of 1s & 0s.

[1 1 0 1 1 0 1 0 1 1]

0 → (-1)

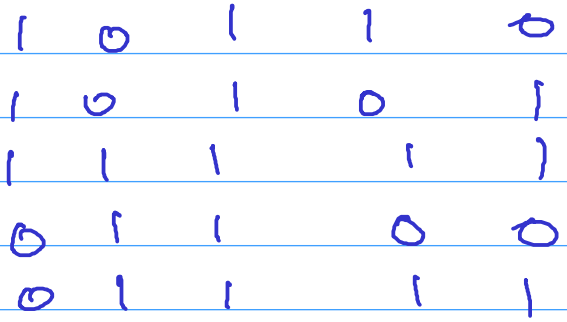
3 → 1s
2 → 0s

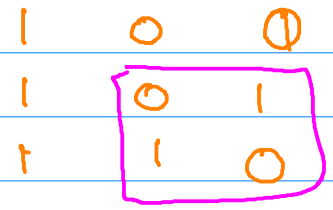
2 1s
1 0s

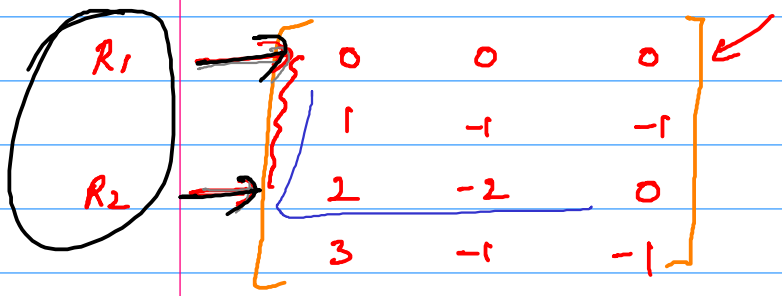
[1 1 -1 1 1 -1 1 -1 1 1]

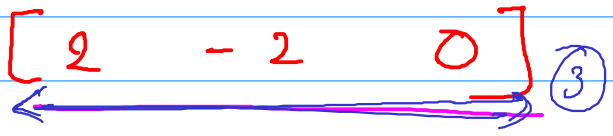
(largest subarray with sum = 0)

Q. Given a binary matrix find largest rectangle in matrix having equal No. of 1s & 0s and return its area.









Maximum length Subarray, Sum = 0)

$3 \times 2 = 6$

Time: $O(N^3)$

Space: $O(N)$ → HashMAP

$1 \le N \le 100$

$1 \le M \le 100$

STACK & Queue:

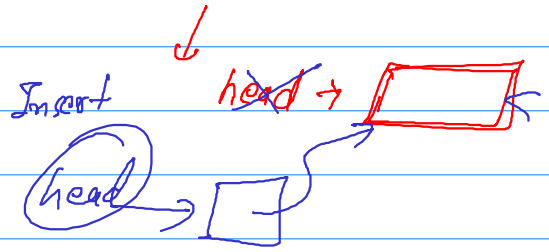
STACK : LIFO

↳ PUSH → O(1)
POP → O(1)
Top → O(1)

Arrays:

① Arrays: ② Linked List ③ 2-Queue

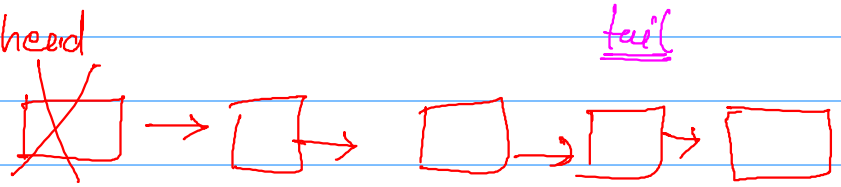
top
Arr



pop head = head.next

Queue

Linked List

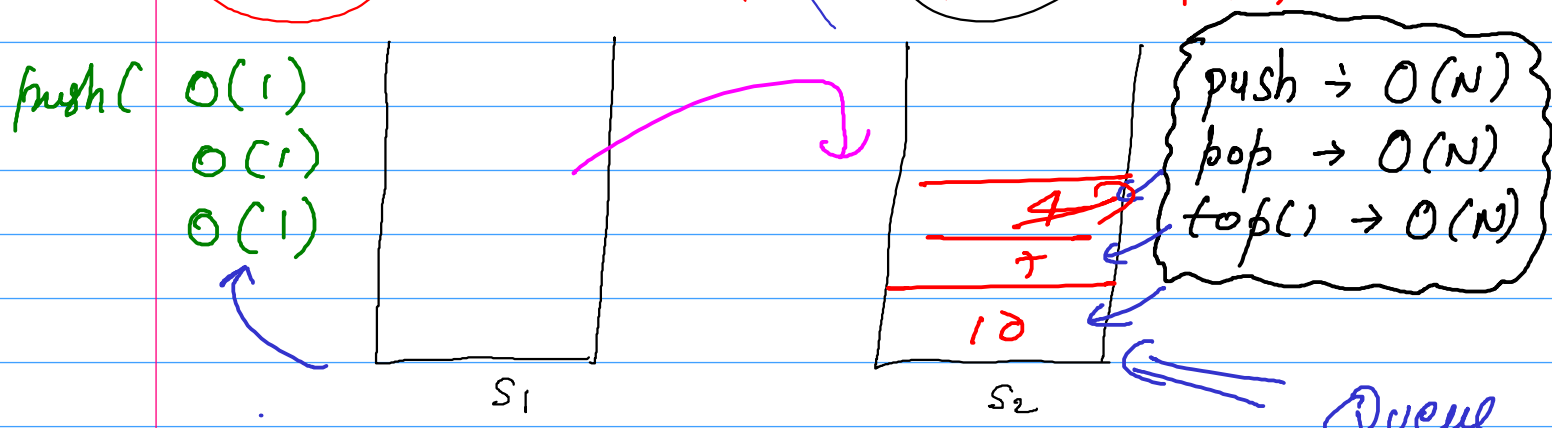


Implement Queue using 2 stacks that support INSERT() DELETE() FRONT() (b/c using any extra variable);

S.poll())

push(1) → push(2) → push(3) → pop() → top() FRONT

push(4) → push(7) → pop() → push(10) → pop() → pop()

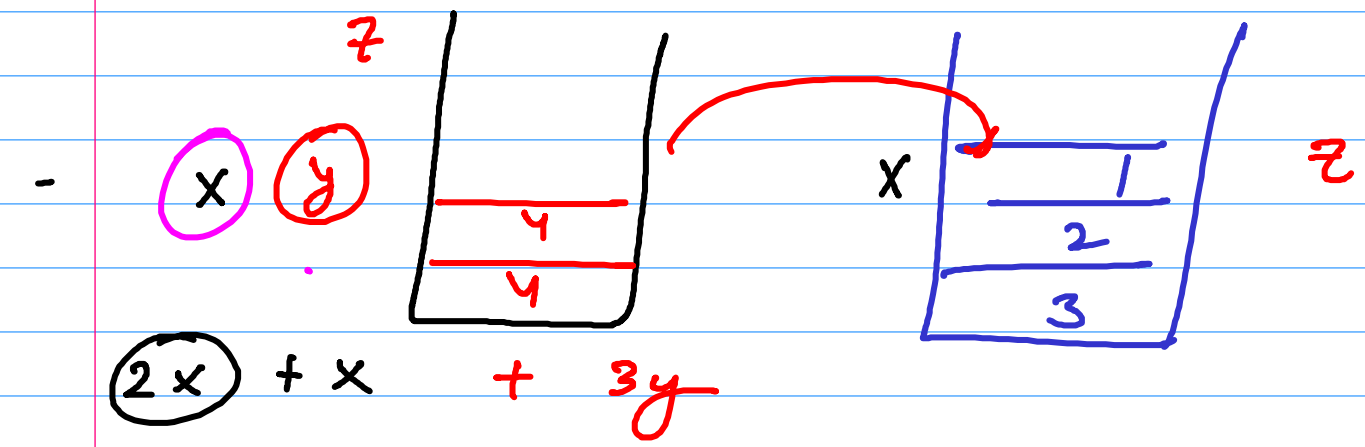


Queue

O(N) → N pop Constant

pop & front: Amortized T.C. O(1)

push O(1)



2x + x + 3y

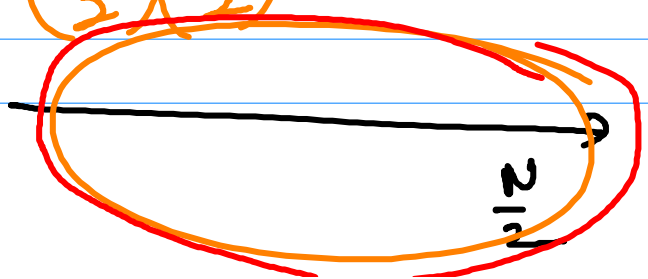
$\frac{N}{2} + \left(\frac{N}{2}\right)\left(\frac{N}{2}\right)$ x + x + x = 3(x)

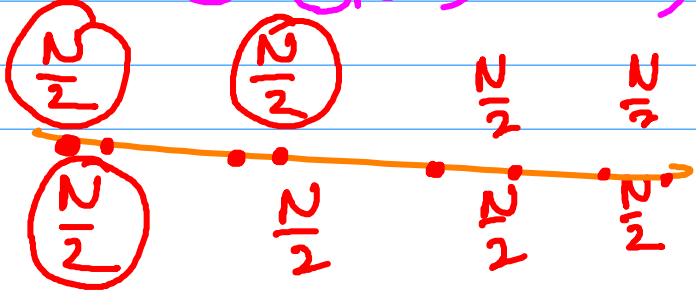
3(4)

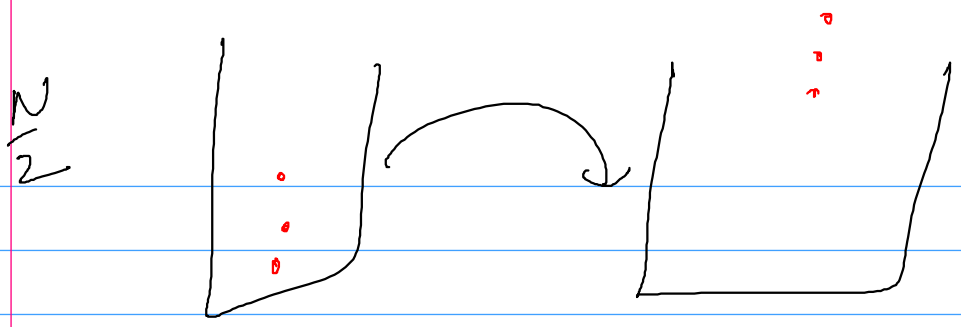
3(2)

$\frac{N}{2}\left(\frac{N}{2} + \frac{N}{4}\right)$ 3(x + y + z + ...)

$\frac{N}{2} + \left(\frac{N}{2}\right)\left(\frac{N}{2}\right)$ 3(N) = O(3N) ≈ O(N)





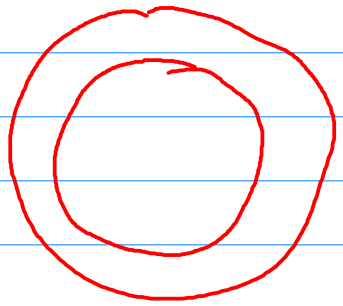


$$+O(1) + O(1) + O(1) + O(1)$$

$$\frac{N}{2} + \frac{N}{2} + \frac{N}{2} + \frac{N}{2}) \simeq O(2N)$$

⊙ Circular Queue:

↓ Array



= x = x = x = y =

```
merge(l, r, A)
if(l>=r) return;
mid = (l+r)/2; //O(1)  O(N)
mer(l, mid, A) -
mer(mid+1, r, A); -
ms(l, mid, r, A); O(N)
}
```

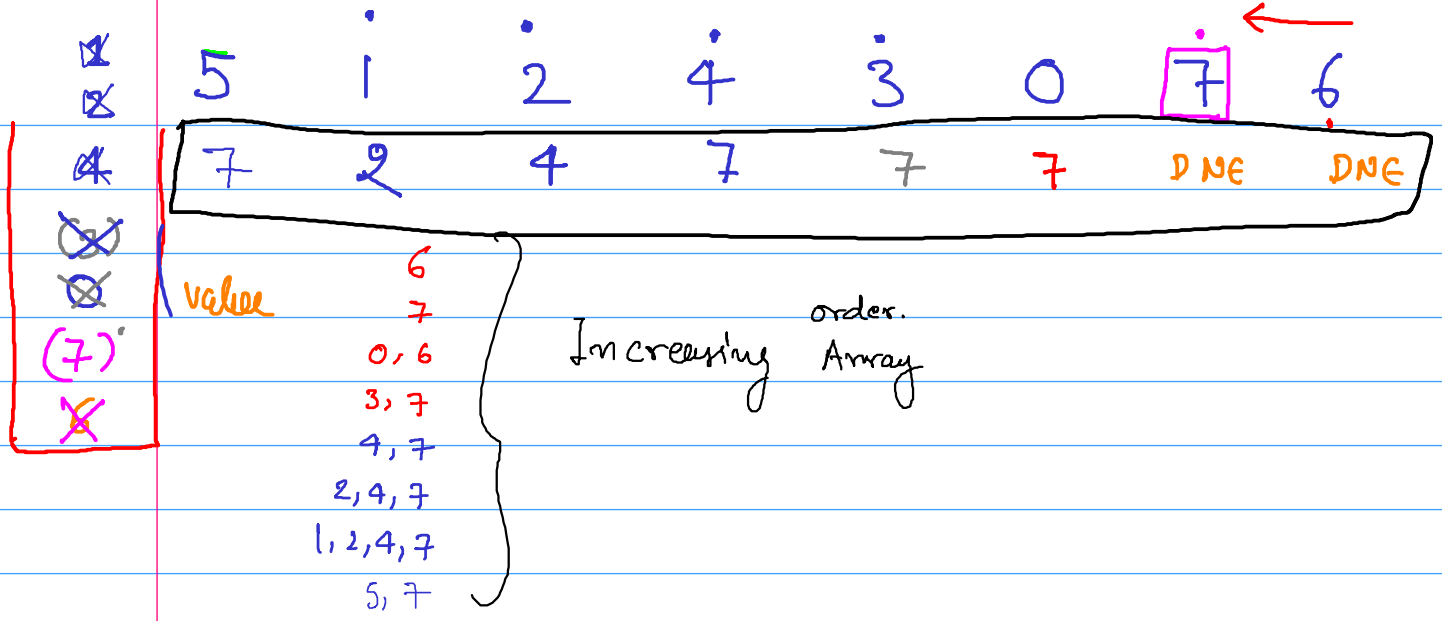
Valid parentheses:

({}, [], ())

( )

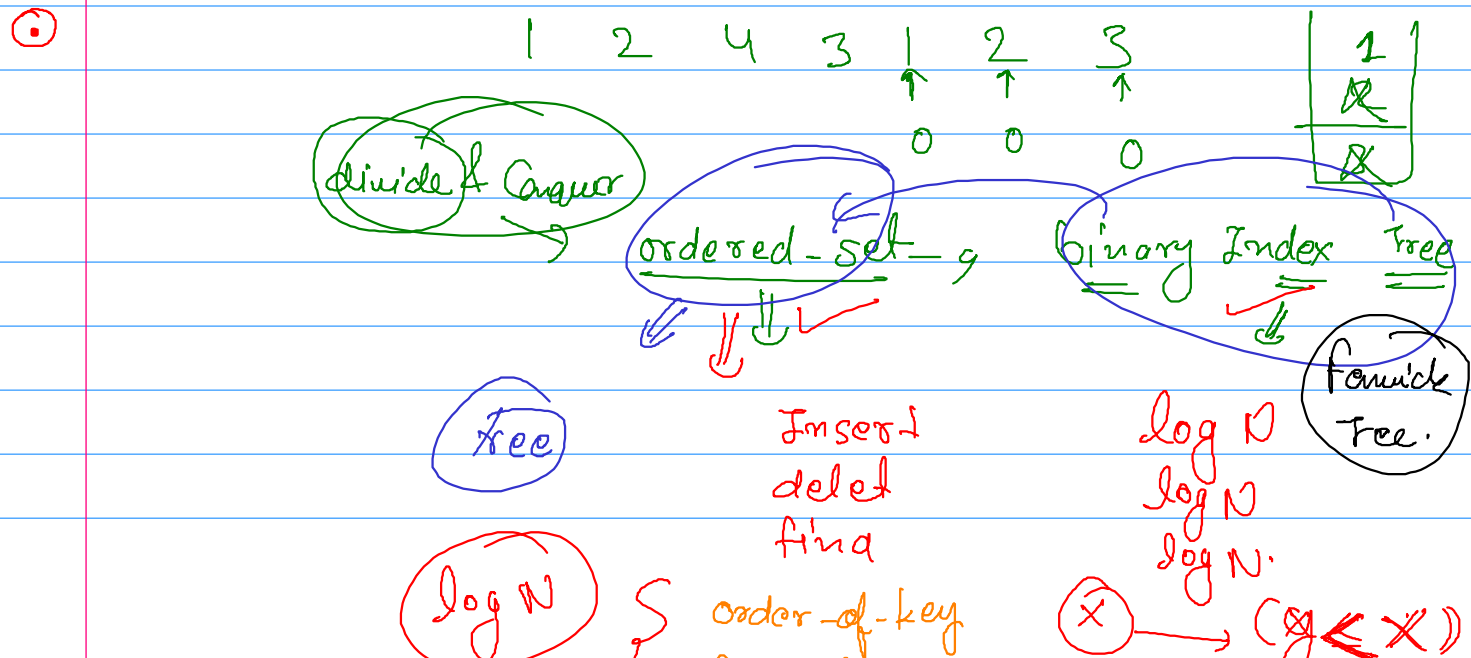
⊙ Next - Greater Element in An array.

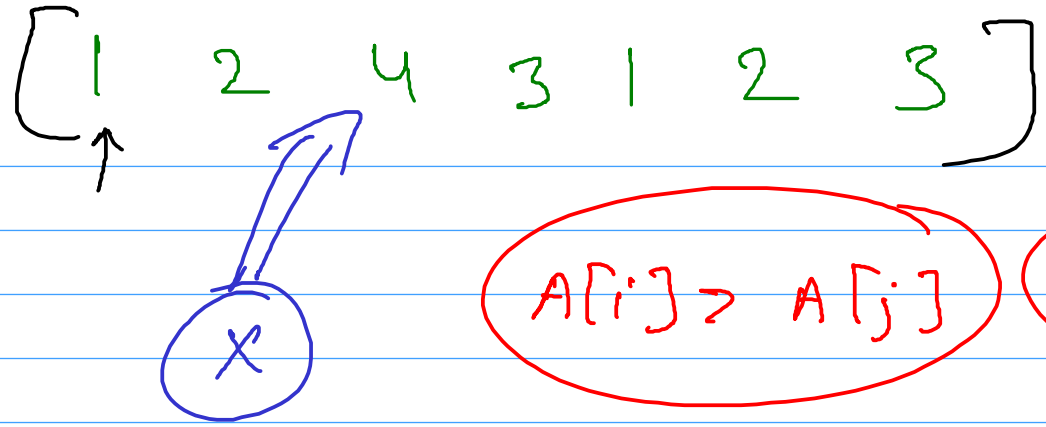
	0	1	2	3	4	5	6	7	8
Array:	5	1	2	4	3	0	7	6	INF
	6 (7)	2 (2)	3 (4)	6 (7)	6 (7)	6 (7)	8 (DNE)	8 (DNE)	




```
Stack<int> S; // Indices.
vector<int> NG(N);
for(int i=n-1; i>=0; i--){
    while(!S.empty() and A[S.top()]<=A[i])
        S.pop();
    if(S.empty()) NG[i]=N;
    else NG[i]=S.top();
    S.push(i);
}
```

Time: O(N)
Space: O(N) — Stack





[1, 2, 4, 3, 1, 2, 3]

[4, 3, 2, 1]

0 1 2 3

ans [1 2 1 0]

[3, 2, 1, 0]

[(4,0) (3,1) (2,2) (1,3)]

[(4,0) (3,1)]

[(2,2) (1,3)]

[(4,0)] [(3,1)]

[(2,2)] [(1,3)]

[(3,1) (4,0)]

[(1,3), (2,2)]

[(1,3), (2,2), (3,1), (4,0)]

(3,1) (4,0) (1,3) (2,2)

i=0 j=0

```cpp
    void merge(int l, int r, vector<pair<int,int>> &arr){
        if(l >= r)return;
        int mid = (l + r) / 2;
        merge(l, mid, arr);
        merge(mid + 1, r, arr);
        Count(l, mid, r, arr);
    }
    vector<int> countSmaller(vector<int>& nums) {
        int n = nums.size();
        ans.resize(n, 0);
        vector<pair<int, int>> arr(n);
        for(int i = 0; i < n ; i++){
            arr[i] = make_pair(nums[i], i);
        }
        merge(0, n - 1, arr);
        return ans;

    }
};
```

```cpp
public:
    vector<int> ans;
    void Count(int l, int mid, int r, vector<pair<int,int>> &arr){
        int sz1 = mid - l + 1;
        int sz2 = r - mid;
        vector<pair<int,int>> a(sz1);
        vector<pair<int,int>> b(sz2);
        for(int i = l; i <= mid ; i++)a[i - l] = arr[i];
        for(int i = mid + 1; i <= r; i++)b[i - (mid + 1)] = arr[i];
        int i = 0, j = 0;
        while(i < sz1){
            if(j < sz2 and a[i].first > b[j].first){
                j++;
            }else{
                ans[a[i].second] += j;
                i++;
            }
        }
        i = 0; j = 0;
        int k = l;
        while(i < sz1 and j < sz2){
            if(a[i].first < b[j].first){
                arr[k++] = a[i++];
            }else{
                arr[k++] = b[j++];
            }
        }
        while(i < sz1)arr[k++] = a[i++];
        while(j < sz2)arr[k++] = b[j++];
    }
```

Description | Solution | Discuss (435) | Submissions

907. Sum of Subarray Minimums

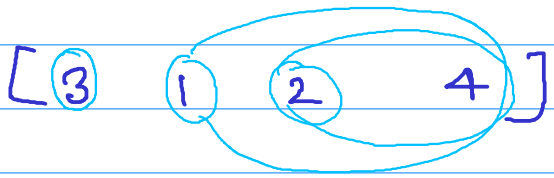
Medium 3448 230 Add to List Share

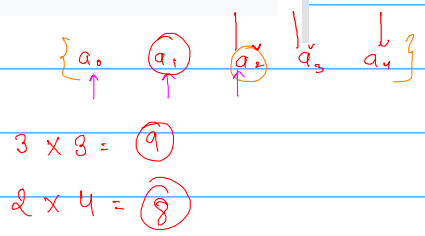
Given an array of integers arr, find the sum of `min(b)`, where `b` ranges over every (contiguous) subarray of `arr`. Since the answer may be large, return the answer **modulo** $10^9 + 7$.

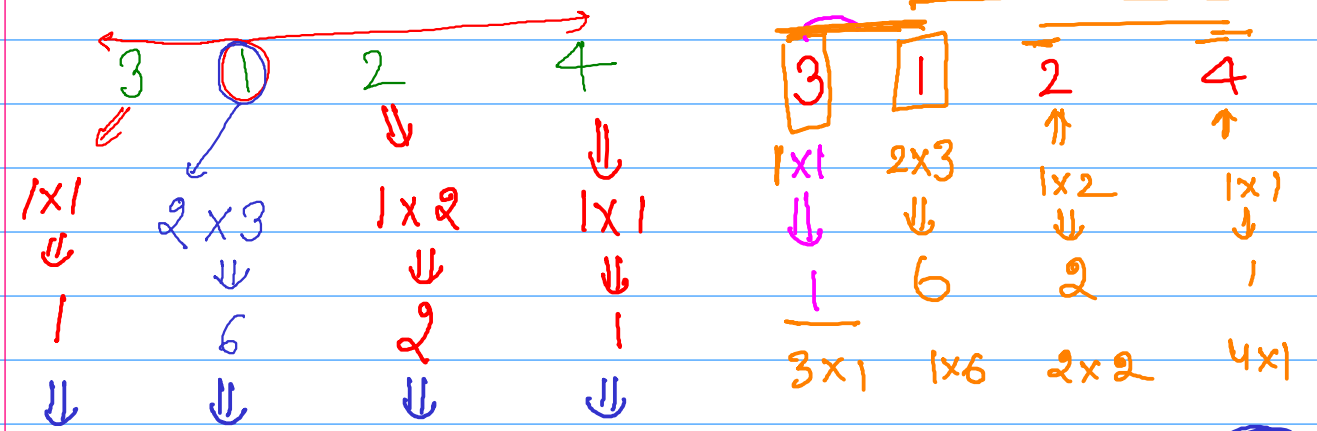
Example 1:

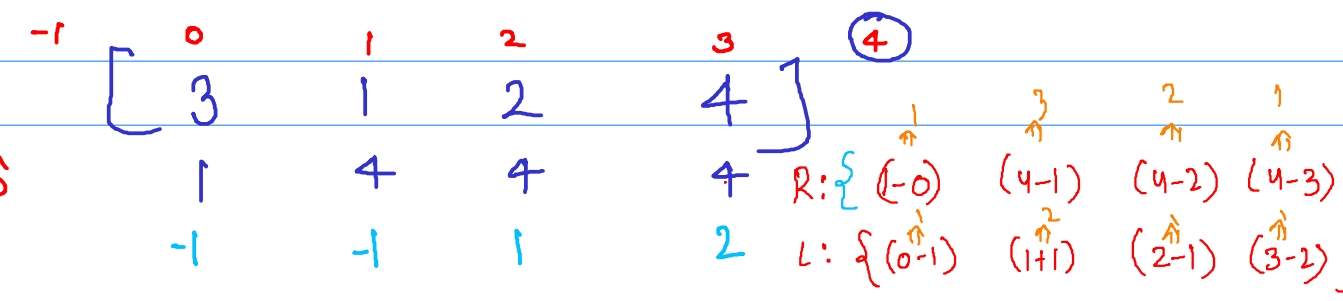
Input: arr = [3,1,2,4]
Output: 17
Explanation:
Subarrays are [3], [1], [2], [4], [3,1], [1,2], [2,4], [3,1,2], [1,2,4], [3,1,2,4].
Minimums are 3, 1, 2, 4, 1, 1, 2, 1, 1, 1.
Sum is 17.

Example 2:









RxL { 1x1 3x2 2x1 1x1 }

3x1 + 1x6 + 2x2 + 4x1

= 3 + 6 + 4 + 4 = 17.

Q.1 Circular Array Next Greater:

1, 2, 1, 1, 2, 1

(2 -1 2) 2 -1 -1

1, 2, +1

[2 -1 2]

1, 2, 1

[1, 2, 3, 5, 3, 3, 5, 5, 1]

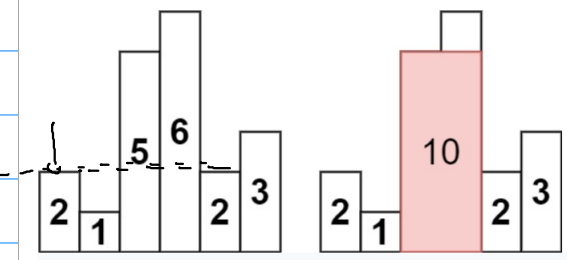
2 3 5 -1 5 5 -1 -1 2

84. Largest Rectangle in Histogram

Hard 9818 144 Add to List Share

Given an array of integers `heights` representing the histogram's bar height where the width of each bar is `1`, return *the area of the largest rectangle in the histogram.*

Example 1:



Area: A[i] · (L − R)

```
Input: heights = [2,1,5,6,2,3]
Output: 10
Explanation: The above is a histogram where width of each bar is 1.
The largest rectangle is shown in the red area, which has an area = 10 units.
```

	-1	0	1	2	3	4	5	6
		2	1	5	6	2	3	
NS		R: 1	6	4	4	6	6	
PS		L: -1	-1	1	2	1	4	
LENGTH (R-L-1)		1	6	2	1	4	1	
		2x1	6x6	5x2	6x1	2x4	3x1	⇒ 10
		(R-1)	(L+1)	10	6	8		
		r	l					

r − l + 1 = R − 1 − L − 1 + 1

= (R − L − 1)

85. Maximal Rectangle

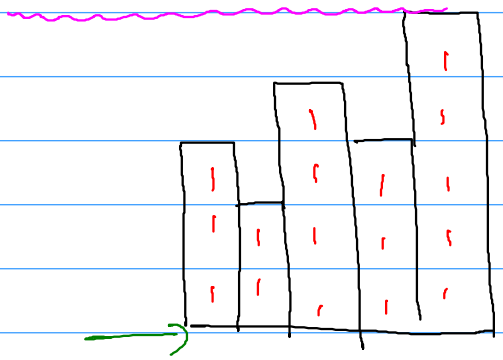
Hard · 6555 · 105 · Add to List · Share

Given a `rows x cols` binary `matrix` filled with `0`'s and `1`'s, find the largest rectangle containing only `1`'s and return *its area*.

Example 1:

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]
Output: 6
Explanation: The maximal rectangle is shown in the above picture.



$R_1 \rightarrow (0 \text{ — } m)$

$R_2 \rightarrow (R_1 \text{ — } m)$

$Ans(0 \text{ — } m)$

$O(n^2 \cdot m)$

Space $(n \times m)$

```
class Solution {
public:
    int maximalRectangle(vector<vector<char>>& mat) {
        int n = mat.size();
        int m = mat[0].size();
        vector<vector<int>> g(n, vector<int>(m, 0));
        for(int i = 0; i < n ; i++){
            for(int j = 0; j < m; j++){
                g[i][j] = (mat[i][j]=='1');

                if(i > 0)g[i][j] += g[i-1][j];
            }
        }
        int ans = 0;
        for(int r1 = 0; r1 < n ; r1++){
            for(int r2 = r1; r2 < n; r2++){
                int ht = r2 - r1 + 1, c = 0;
                for(int j = 0; j < m; j++){
                    int sum = g[r2][j];
                    if(r1 > 0)
                        sum -= g[r1 - 1][j];
                    if(sum == ht)c++;
                    else c = 0;
                    ans = max(ans, ht * c);
                }
            }
        }
        return ans;
    }
};
```

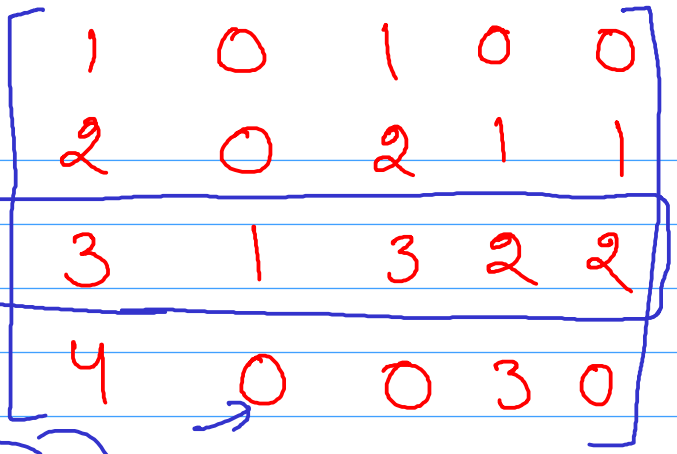
Time: $O(n^2 \cdot m)$

Space $O(n \cdot m)$

$g[r_2][0] - g[r_1-1][0]$

Example 1:

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0



O(n x m)

O(n x m)

O(n x m)

```cpp
    int maximalRectangle(vector<vector<char>>& mat) {
        int n = mat.size(), ans= 0;
        int m = mat[0].size();
        vector<int> row(m, 0);
        for(int i = 0; i < n ; i++){
            for(int j = 0; j < m; j++){
                if(mat[i][j] == '1'){
                    row[j]++;
                }else{
                    row[j] = 0;
                }
            }
            ans =  max(ans, getans(row));
        }
        return ans;
    }
};
```

```cpp
    int getans(vector<int> &arr){
        int n = arr.size();
        vector<int> ns(n, n), ps(n, -1);
        stack<int> s;
        for(int i = n - 1; i >= 0; i--){
            while(!s.empty() and arr[s.top()] >= arr[i])
                s.pop();
            if(!s.empty())ns[i] = s.top();
            s.push(i);
        }
        while(!s.empty())s.pop();
        for(int i = 0; i < n ; i++){
            while(!s.empty() and arr[s.top()] >= arr[i])
                s.pop();
            if(!s.empty())ps[i] = s.top();
            s.push(i);
        }
        int ans = 0;
        for(int i = 0; i < n ; i++){
            int l = ps[i], r = ns[i];
            int ar = arr[i] * (r - l - 1);
            ans =max(ans, ar);
        }
        return ans;
    }
```

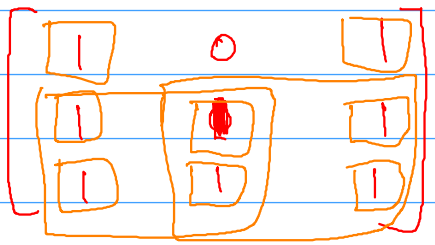
Time: O(n x (m + m))

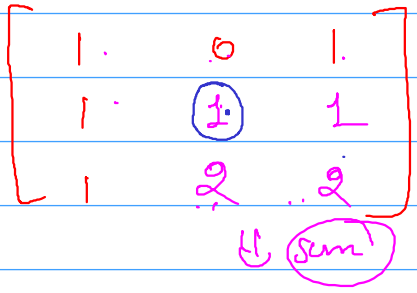
pre ans cal.

Space: (n + n + n + n)

row ns ps stack

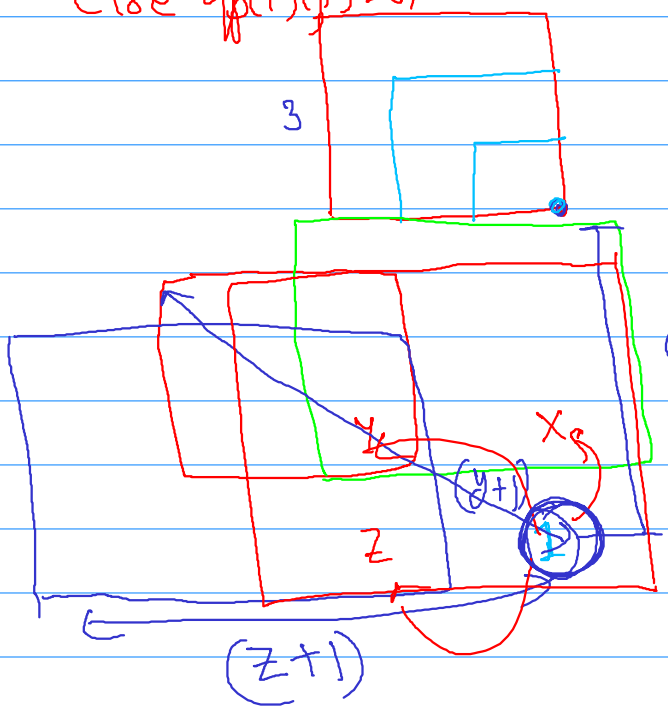
⑩ Count. No. of Square Submatrices containg all ones in a binary matrix.





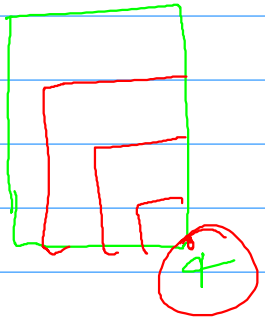
if(M[i][j]==1) dp[i][j] = 1 + min(dp[i-1][j-1], min(dp[i-1][j], dp[i][j-1]));

else dp[i][j] = 0;



min(x+1, y+1, z+1)

1 + min(x, y, z)



0 on .. No. of rectangular Submatrices having all ones in a binary matrix.

$1 \le N, M \le 100$

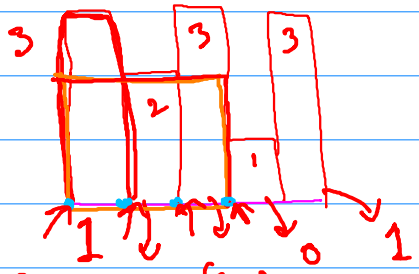
$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$O(N \cdot N \cdot M)$ 10^6

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 2 & 3 & 2 \end{bmatrix}$$

$1 \le N, M \le 1000$

10^9 op

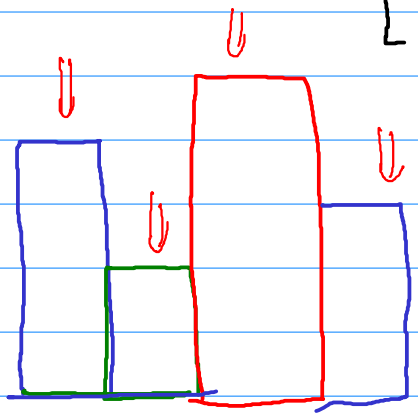


$^4C_2$:

$\frac{4 \times 3}{2} = \left(\frac{3 \times (3+1)}{2}\right)$ (1+1) (2+1)

$= 7$

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



[1, 2, 5, 3]

155. Min Stack

Easy 7969 617 Add to List Share

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the `MinStack` class:

- `MinStack()` initializes the stack object.
- `void push(int val)` pushes the element `val` onto the stack.
- `void pop()` removes the element on the top of the stack.
- `int top()` gets the top element of the stack.
- `int getMin()` retrieves the minimum element in the stack.

Example 1:

```
Input
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]
```

stack<int> S;

stack<int> MinS;

S.push(val);
MinS.push(min(x, val));
x = min(x, val);

getMin(){
return x;

pop(){
S.pop();
if(min == MinS.pop())
x = min(x,
x = S.top();

Using 2 stack.

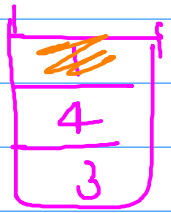
int x = -INF;
push(val);
{ x = min(x, val);
S.push(val);
MinS.push(x);
}

getMin(){
return x;
}.

pop(){
S.pop();
MinS.pop();
x = MinS.top();
}

3 4 1 pop

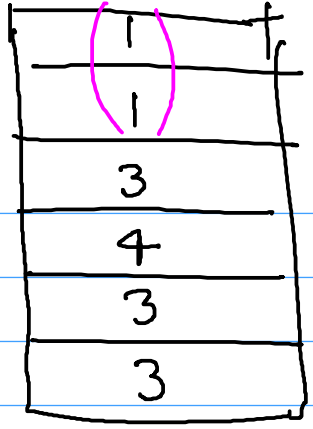
pop()



Min = 1

Min = 3

3 4 1



pair<int, int>

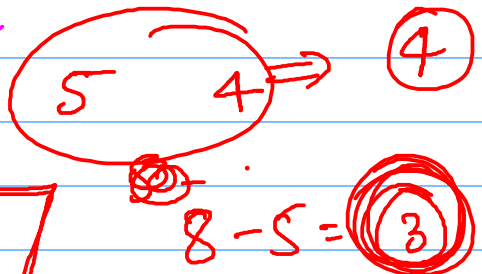
int MinE = ;

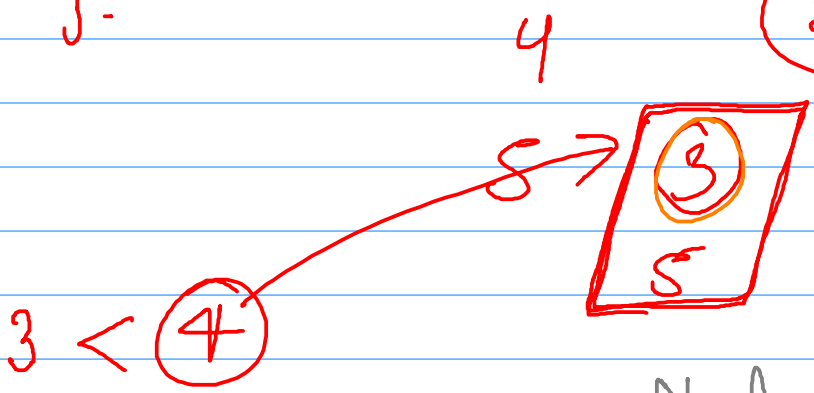
```
push(x) {
    if (s.empty()) {
        st.push(x);
        MinE = x;
    }
    else{
        if(x >= MinE) {
            s.push(x);
        }
        else{
            int Nval = 2x - minE;
            s.push(Nval);
            MinE = x;
        }
    }
```

MinE > X

MinE + x > 2x

MinE > X > 2x - minE.





Nval = 2 * X - MinE.

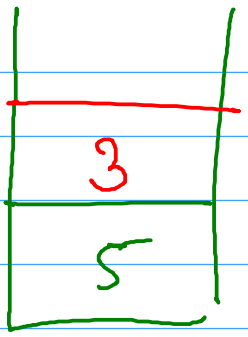
```
getMin()
    return MinE;
```

```
top() {
    if(s.top() < MinE)
        return MinE;
    } return s.top();
```

s.top()

5, 4

MinE = 5



= 2*4 - 3

= 8 - 3 = 5

8 - 5 = 3

MinE = 4

getMin()
return minE;

top() {
if (S.top() < MinE)
return MinE;
return S.top();

pop() {
if (S.top() < MinE) {
minE = 2 * MinE - S.top() }
}
S.pop();

Nval = 2 · x - minVal.

minE = 2 * x - Nval

minE S.top()

2 * MinE - S.top()

Min = 2x - Nval

Min S.top()

Min > x

Min + x > x + x

Min + x > 2x

x > 2x - Min

Nval = 2x - Min

minE > x

Nval < Min ← Nval = 2*x - Min

S.push(Nval)

Min = x;

Nval = 2·x - Min

Min = 2x - Nval

```
pop() {
  if (s.top() < Min) {
    Min = 2·Min - s.top()
  }
  s.pop();
}
```