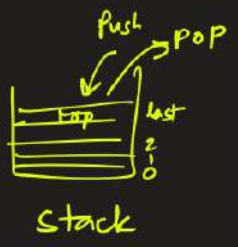
Stack, Queues, Priority Queues & Heaps Lecture 1

Sunday, 4 August 2024 2:03 PM

Stacks



stack

C++

<stack.h>

stack<int> s;
<T>

s.push(_) O(1) ✓

s.pop(); O(1) ✓

s.top() → top element O(1) ✓

s.empty() ← True / False

s.size() → ✓

Python

from collections import deque

s = deque();

✓ s.append(_);

✓ s.pop();

✓ s[-1]; ← top

✓ if s: → True (Not empty) / False
bool(s)

✓ len(s)

https://leetcode.com/problems/valid-parentheses/

() valid

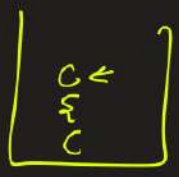
({}) valid

([}) Invalid

(Invalid

(, {, [,), },]

eg: (({ })) [ ]



({ ({ })) [ ]

Invalid

open – push

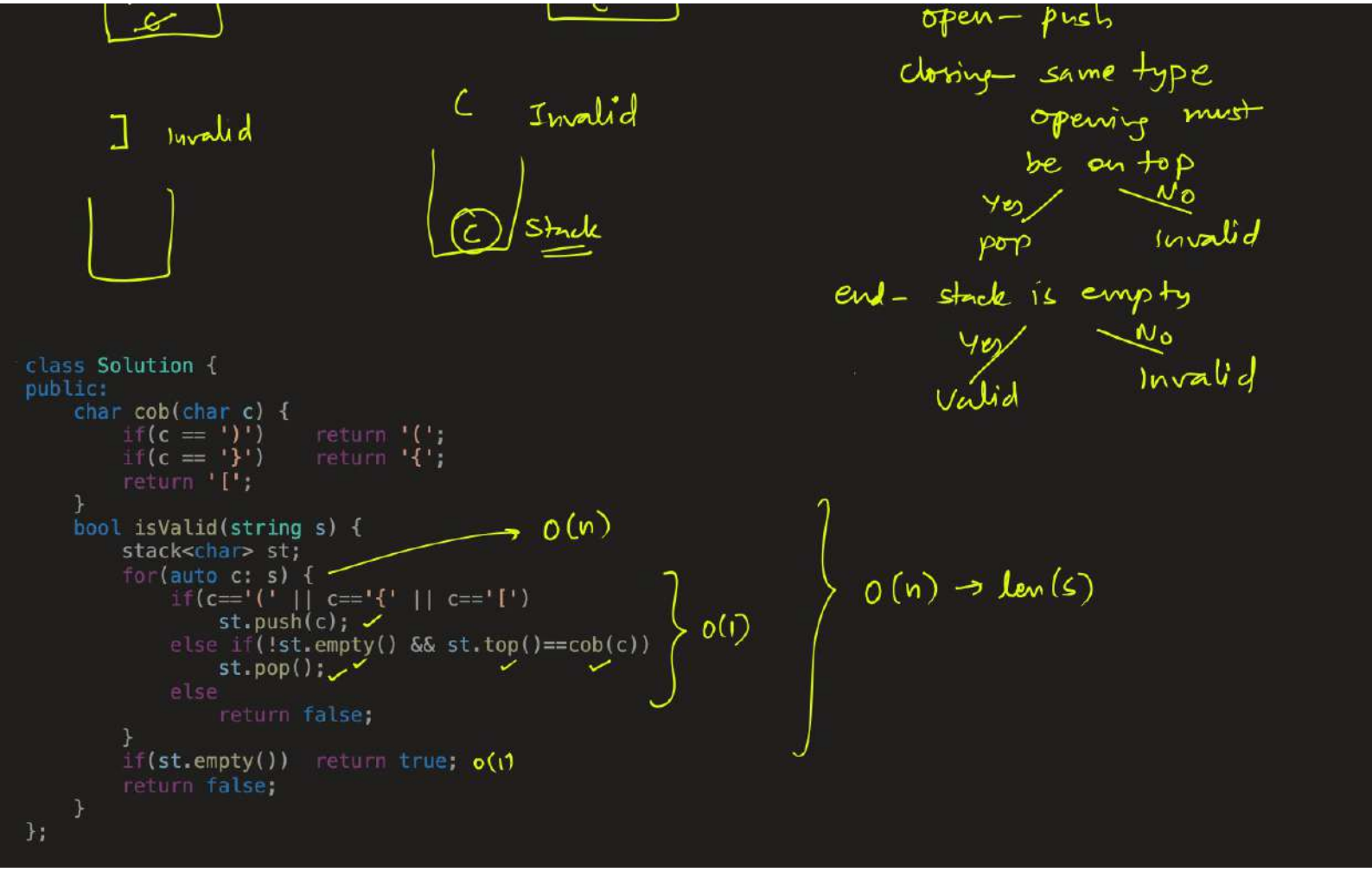
closing – same type opening must be on top

Yes / pop

No / invalid

] Invalid

(Invalid


open – push
closing – same type
opening must
be on top
Yes
No
pop
Invalid
end – stack is empty
Yes
No
Valid
Invalid
] Invalid
(Invalid
Stack
class Solution {
public:
char cob(char c) {
if(c == ')') return '(';
if(c == '}') return '{';
return '[';
}
bool isValid(string s) {
stack<char> st;
for(auto c: s) {
if(c=='(' || c=='{' || c=='[')
st.push(c);
else if(!st.empty() && st.top()==cob(c))
st.pop();
else
return false;
}
if(st.empty()) return true;
return false;
}
};
O(n)
O(1)
O(n) → len(s)

Next greater Element

Given an array, find the index/element of next greater element for all elements in the array.

```
      0  1  2  3
     [1, 3, 4, 2]
      ↑
```

Next gr. elem [3, 4, -1, -1]

next gr index [1, 2, -1, -1]

[5, 2, 3, 4, 6, 7]

[6, 3, 4, 6, 7, -1]

Brute :- $O(n^2)$ ✓

Stacks

6, 4, 6, 6, 7, -1

stack.top()

e → { t <= e pop()
    { t > e ← ans.

✓ problem | traversa

stack.top()

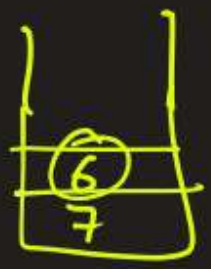
e → { t <= e pop()
    { t > e ← ans.

✓ problem		traversal	pop	ans
right	greater ✓	reverse	$t \le e$	~~$t > e$~~
right	lesser ✓	reverse	$t \ge e$	$t < e$
left	greater ✓	same	$t \le e$	$t > e$
left	lesser	same	$t \ge e$	$t < e$

Stacks

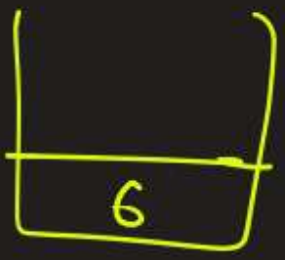
[5, 2, 4, 3, 6, 7]

6, 4, 6, 6, 7, -1

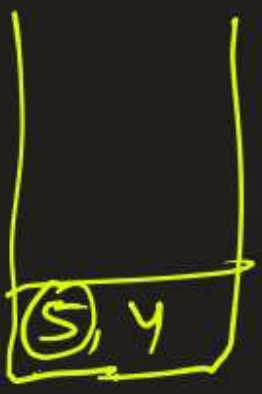


[3, 6, 2, 5, 1, 4]

[6, -1, 5, -1, 4, -1]



	0	1	2	3	4	5	6	7
	4	2	1	3	5	4	3	∞
elem	5	3	3	5	-1	-1	-1	
index	4	3	3	4	7	7	7	



https://leetcode.com/problems/next-greater-element-i/

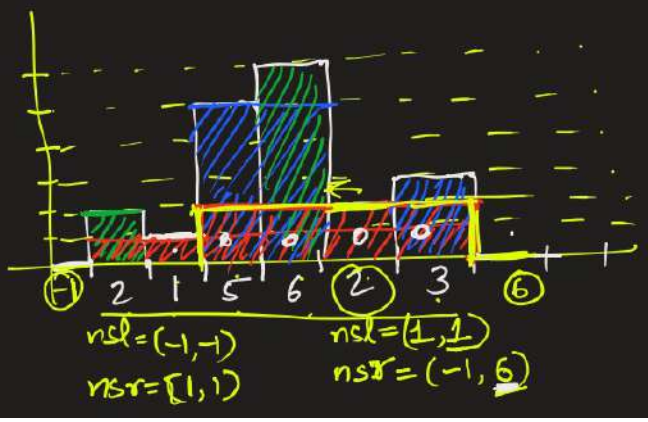
```
from collections import deque

class Solution:
    def nextGreaterElement(self, nums1: List[int], nums2: List[int]) -> List[int]:
        s = deque()
        d = {}
        for i in range(len(nums2)-1, -1, -1):
            while s and s[-1]<=nums2[i]:
                s.pop()
            if s:
                d[nums2[i]] = s[-1]
            else:
                d[nums2[i]] = -1
            s.append(nums2[i])
        return [d[n] for n in nums1]
```

Largest Rectangular Area in Histogram

https://leetcode.com/problems/largest-rectangle-in-histogram/

eg [2, 1, 5, 6, 2, 3]



Rectangle

(2) arr[0] :- 2

(1) arr[1] :- 6

(5) arr[2] :- 10 (10)

(6) arr[3] :- 6

(2) arr[4] :- 8

(3) arr[5] :- 3

for any index i

↳ Find the rectangle with arr[i] as the height

↳ How far can you go left — O(n)

↳ nse to the left (indx nsl)

↳ How far can you go right — O(n)

↳ nse to the right (indx nsr)

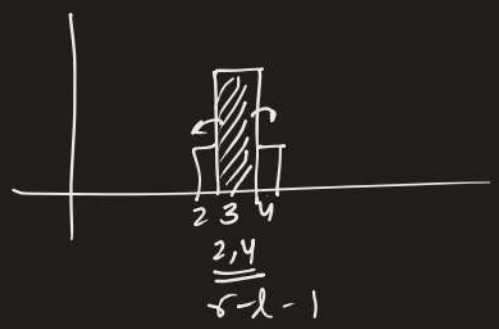
↳ size of rectangle =

arr[i] * (nsr - nsl - 1)

height width

Time:- O(n)

space:- O(n)



right: 4 3 2 1 0 -1

1, 2, 3, 4, 5

left: -1 0 1 2 3 4

-1 0 1 2 3 4 5

size-ind-1

4 → 0
3 → 1
2 → 2
1 → 3
0 → 4
-1 → 5

size-ind-1
5-4-1

	5	4	3	2	1	0	
	0	1	2	3	4	5	6
	2	1	5	6	2	3	
l	-1	-1	1	2	1	4	6
r	4	-1	1	1	-1	-1	
	1	6	4	4	6	6	

(size - r[i] - 1 - l[i] - 1)

1 - (-1) - 1 = 1

4 - 1 - 1
= 2

```python
from collections import deque
def nsel(nums):
    s = deque()
    ans = []
    for i, n in enumerate(nums):
        while s and s[-1][0]>=n:
            s.pop()
        if s:
            ans.append(s[-1])
        else:
            ans.append((-1, -1))
        s.append((n, i))
    return ans
class Solution:
    def largestRectangleArea(self, heights: List[int]) -> int:
        l = nsel(heights)
        r = nsel(heights[::-1])
        r.reverse()
        ans = 0
        for i, h in enumerate(heights):
            val = h*(len(heights)-r[i][1]-l[i][1]-2)
            ans = max(ans, val)
        return ans
```

```cpp
class Solution {
public:
    vector<pair<int, int>> nsel(vector<int> &nums) {
        vector<pair<int, int>> ans;
        stack<pair<int, int>> s;
        for(int i=0; i<nums.size(); i++) {
            while(!s.empty() && s.top().first >= nums[i]) s.pop();
            if(s.empty()) ans.push_back(make_pair(-1, -1));
            else ans.push_back(s.top());
            s.push(make_pair(nums[i], i));
        }
        return ans;
    }
    int largestRectangleArea(vector<int>& heights) {
        vector<int> rHeights;
        for(int i=heights.size()-1; i>=0; i--)  rHeights.push_back(heights[i]);
        vector<pair<int, int>> l = nsel(heights), r=nsel(rHeights);
        reverse(r.begin(), r.end());
        int ans = 0;
        for(int i=0; i<heights.size(); i++) {
            int val = heights[i]*(heights.size()-r[i].second-l[i].second-2);
            ans = max(ans, val);
        }
        return ans;
    }
};
```

https://www.geeksforgeeks.org/problems/restrictive-candy-crush--141631/1

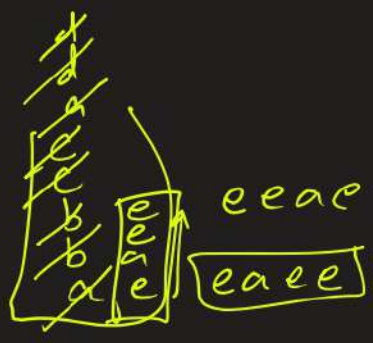
k = 3

a b b c c c b a d d d a e a e e

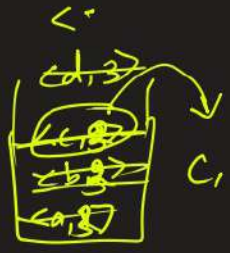
↳ a b b b a a e a e e

↳ a a a e a e e.

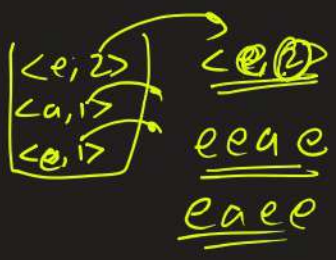
↳ e a e e



a b b c c c b a d d d a e a e e



c, 3 = k



```
class Solution{
    public:
    string Reduced_String(int k,string s){
        // Your code goes here
        stack<pair<char, int>> st;
        for(auto c: s) {
            if(!st.empty() && st.top().first == c) {
                pair<char, int> p = st.top();
                p.second++;
                st.pop();
                st.push(p);
            }
            else
                st.push(make_pair(c, 1));
            if(!st.empty() && st.top().second == k)
                st.pop();
        }
        string ans;
        while(!st.empty()) {
            pair<char, int> p = st.top();
            while(p.second--) ans += p.first;
            st.pop();
        }
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```