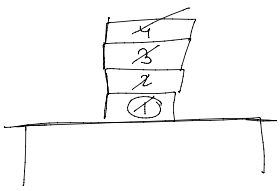
03 June 2024 21:40

Stacks & Queues

Stacks

FILO → First in Last out



4, 3, 2, 1

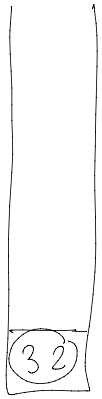
push : add an element.
pop : remove an element
peek : see the top element.

Time complexity
O(1)

Evaluate reverse polish notation

2 + 3 * 10 = 32

2 3 10 * +



3 × 10 = 30

30 + 2 = 32

4 + 2 → 4 2 +

3 − 6 → 3 6 −

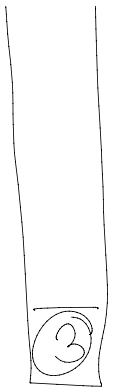
4 2 + 3 =

4 + 2 = 2 + 4

4 − 2 ≠ 2 − 4

6 − 3

4 2 + 3 =



4 + 2 = 6

6 - 3

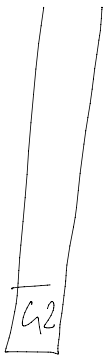
3 - 6

6 - 3

6 - 3

4 + 5 * 4 + 6 * 3

4 5 4 * + 6 3 * +



42

42

5 10 2 / +

10/2

1) Creati a stack.

2) Iterati through the input.

3) If it is an operand, push it into the stack.

4) If it is an operator:

int num1 = stack top;

4) else

int num1 = stack top;

stack pop;

int num2 = stack top;

stack pop;

stack push(num2 operator num1)

5) return stack top;

Nearest smaller element

[4, 5, 2, 10, 8]

[-1, 4, -1, 2, 2]

[1, 2, 3]

[-1, 1, 1] min=1

Brute force $O(n^2)$

```
int arr[A.size()] = {-1};
for(int i=0; i<A.size(); i++){
    for(int j=i-1; j>=0; j--){
        if(A[j]<A[i]){
            arr[i] = A[j];
            break;
        }
    }
}
return arr;
```

i: [4, 5, 2, 10, 8] (0 1 2 3 4)

[-1, 4, -1, 2, 2]

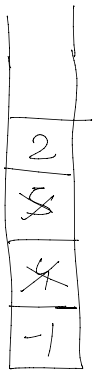
[9, 3, 5, 6, 4]

⇒ [-1, -1, 3, 5, 3]

[4, 5, 2, 10, 8]

[-1, 4, -1, 2, 2]

T: O(n)
S: O(n)



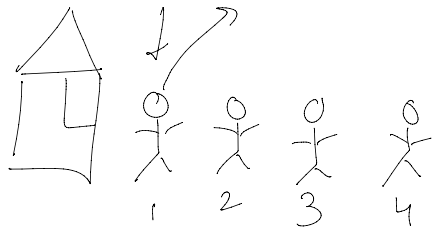
- Remove all the elements ≥ current element
- Put stack top into the ans.
- Put the current element into the stack.

create a stack, put -1 into it.

```
for(int i=0; i<A.size(); i++) {
        while(stack top >= A[i])
                stack pop;

        ans[i] = stack top
        stack push(A[i]);
}
return ans;
```

Queue



FIFO
First in first out

1) Enqueue : Add element to queue → Time complexity O(1)

1) Enqueue : Add element to queue
2) Deque : Remove element from queue
3) front : See the front element

Time complexity → O(1)

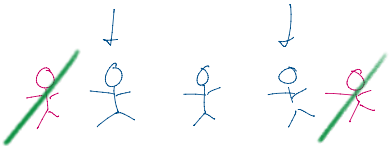
Sliding window maximum

[1, 3, -1, -3, 5, 3, 6, 7] K = 3

[3, 3, 5, 5, 6, 7]

K

Double ended queue (Stack + queue) :



1) Add front
2) Add back
3) Remove front
4) Remove back
5) front
6) last

O(1)

[1, 3, -1, -3, 5, 3, 6, 7]

- remove all elements less than current element from back.
- remove all element out of window from the front
- Add current element at the back.

→ [6, 7]

[3, 3, 5, 5, 6, 7]

maintain a queue with indices instead of values.

T: $O(N)$ S: $O(N)$

Reverse first k elements of a queue

[1, 2, 3, 4, 5] k = 3 $\Rightarrow$ [3, 2, 1, 4, 5]

[3, 2, 1, 4, 5]