Stack Introduction and Identification

- Questions are interconnected to each other.
- Stack is important from interview point of view.
- we can write brute force approach, it is important to identify which data structure will decrease its time complexity to O(Best).

1. Nearest greater to Left
2. Nearst greater to Right
3. Nearest Smaller to Left
4. Nearest Smaller to Right
5. Stock span problem
6. Maximum Area of Histogram.
7. Max Area of Rectangle in Binary Matrix.
8. Rain water Trapping
9. Implementing a min stack ~ Extra space
10. Implementing stack using Heap
11. longest valid Parenthesis.
12. Iterative TOH.

IDENTIFICATION

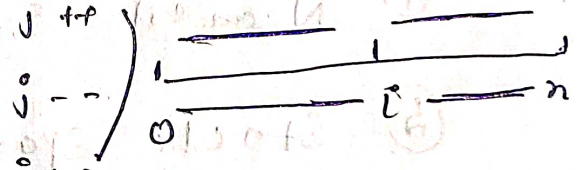
- If question is of Array then there is probability to apply **stack**.
- When we had written a brute force approach $O(n^2)$ with two loop, and in another loop there is some relation b/w the 1st loop then we can apply stack for better Complexity

$j = function(i) \Rightarrow$ **stack**.

- j loop is dependent on i the stack can better Complexity

```
for (int i=0; i<n; i++)
```

for (j → 0 to i j++
      j → i to 0 j--
      j → i to n j++
      j → n to i j--)



We can apply **stack**.

↳ Question
↳ Identification
↳ Concepts
  ↳ Variations in Concepts to solve New **problem**.

- **Pattern**
  +
- **Variation**

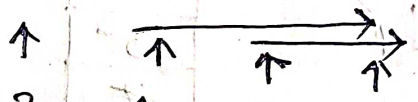
Nearest Greater to right / Next Largest Element

Problem statement

Given an array, print the next Greater element (NGE) for every element. The next Greater element for an element x is the 1st Greater element on the right side of x in the array. Element for which no greater element exist, Consider next greater element as -1.

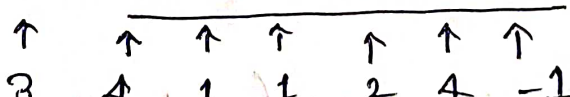
example.

arr[]: 1 3 2 4



3 4 4 -1.

arr[]: 1 3 0 0 1 2 4



3 4 1 1 2 4 -1.

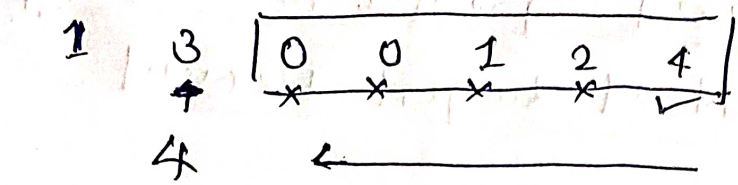
Brute force.

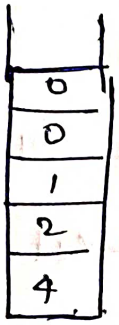
i = 0 ----

j = i+1 ----

```
for(i=0; i<n-1; i++)
```

j depends on i ⇐ `for(j=i+1; j<n; j++)`





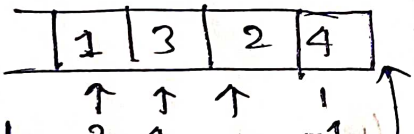
Stack.

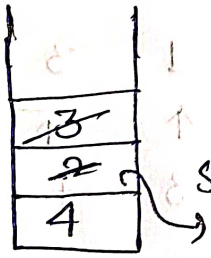
LIFO

- 0 No pop it.
- 0 No pop it.
- 1 Not greater than 3 pop it
- 2 " " " 3 pop it
- 4 Yes it is greater take it.

↳ If Stack became empty return -1

Example





vector v: -1 4 4 3

Now reverse vector.

reverse (v.begin(), v.end());

return v;

$O(n^2) \rightarrow O(n)$

What we did:-

if Stack is Empty() ⟶ -1 (Push in vector)

S.top() > arr[i] → push S.top() in vector

S.top() <= arr[i] → pop
- Stack Empty up to when S.top() greater than arr[i]

Code:

```
Vector<int> v;
Stack<int> s;
// for traversing right to left.
for(int i= size-1; i>=0; i--)
{
  // Cond for checking either stack is empty
  if(s.size()==0)
      v.push_back(-1);
  else if(s.size()>0 && s.top()>arr[i])
  {
     v.push_back(s.top())
  }
  else if(s.size()>0 && s.top()<=arr[i])
  {
     while(s.size()>0 && s.top()<=arr[i]
     {
        s.pop();
```

```
        if ( S.size() == 0 )
        {
            v.push_back(-1)
        }
        else
        {
            v.push_back(S.top())
        }
        S.push_back(arr[i]);
    }
    return vector v;
}
```

:- This is base code for all other questions with little variation.