1.NEAREST GREATER ELEMENT TO RIGHT:
Use stack in questions,in which brute force has two nested loops in which ‘j’ i.e.of inside loop depends on ‘i’.(O(n*n) → O(n))

For i:n-1 to 0 → till st.top()<=arr[i],pop out elements from stack.If stack.empty() ngr[i]=-1 else ngr[i]=st.top();
Push arr[i] for every i.

Since for an element ,its nearest right element should be at the top of the stack,we calculate NGE from right to left,because we keep pushing all elements from right to left.
n0,n1 n2 n3 n4 ...
Let us say we have a stack in which n5,n4,n3 are there.(n3 at top)
If n4 <=n2 ,we can pop out n4 ,n4 will never be the next greater element for any element,left to n2.
Because if n1<n2,n2 will be NGE for n1,if n1>=n2,n4<=n1 also no problem in popping out.

Variations:NEAREST GREATER ELEMENT TO LEFT,NEAREST SMALLER ELEMENT TO RIGHT,NEAREST GREATER ELEMENT TO LEFT

Other questions:Stock span,area of histogram,max area rectangle in binary matrix(use are of histogram concept).

2.Celebrity problem:

First think why there can be at most one celebrity.

If we take two persons and analyze their relationship,we can easily eliminate at least one as a celebrity.If both knows each other or both doesn’t know each other,both will certainly not be celebrity in this case. If only one knows another,then the other one can be a celebrity.

Take a stack and insert all no.s (0..n-1) into it.The stack represents potential celebrities.
Pop out two elements from stack and analyze their mutual relationship.And if there is,insert back that
no. which can be a celebrity.
Do this until stack size!=1 or !=0.
If 0,no celebrity.If 1,then the no. can be celebrity.Check its corresponding column and row in matrix if it can be celebrity or not.

LRU Cache

Brute:If we use array ,then push() will take O(n) time because sometimes we have to delete from array.
We have to delete a no.in O(1) time ,this gives us intuition that we should use a double linked list.

Our LRU node will be next to head.Therefore while push(),we will push the node to next of head.And while get(key),key will become our LRU,so we will delete the node from there and push again to next of the head.
Since we want to access a random node by key in O(1) time and not possible in DLL(adding recent to head->next and Least frequently to tail->prev),we get intiution to use map for this.

Queue implementation using array:

Two special cases:
1.Queue is empty when rear==-1||front>rear
2.When front==-1,while inserting a no.,front=0;

Rest all are the same as stack cases.
Array implementation of queue wastes a lot of space that's why use linked list.

In linked list,only change is in insertion (just like array implementation),rest all that of stack

When we want the topmost element but want to delete a random element from the list,use DLL.

Sorting stack using recursion such that greatest element at top:

Sort function:sort(st)=place(sort.top(),sort(st.pop())
Place function: Place(a,st) function will place a no. ‘a’ in a sorted stack. If a>st.top(),we can simply push ‘a’ into the stack else place(a,st)=(place(a,st.pop()))).push(st.top()) //understand it how.

Interconversion of postfix,prefix and infix expression:
Study later.

Sliding window maximum

M1. using priority queue
M2: using dequeu