Stack

Concept - I

1. Nearest greater to left / right
2. " Smaller " "
3. Stock span problem
4. maximum area of Histogram
5. maximum area of rectangle in Banary search
6. Rain water trapping
7. implementating a main Stack
8. implementating stack using heap
9. the celebrity Problem
10. longest valid parenthesis
11. Iterative TOH.

Concept II

Identification

I. Array < Heap / Stack
Stack

II $O(n^2)$

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

यहाँ i; j पर depend नहीं कर रहा है but in stack question. i, j पर depend करेगा।

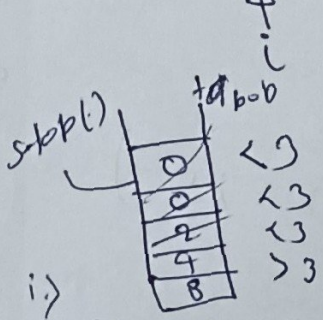
ex.

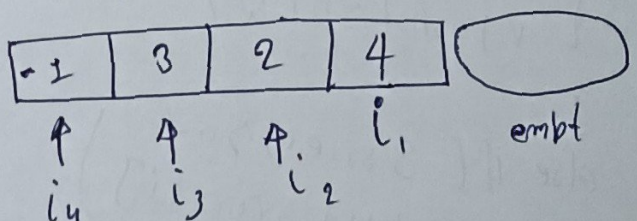
```
for (int i=0; i<n; i++)
  for( int j=0; j<i; j++)
```

for (int i=0; i<n-1; i++)
for (int j=i+1; j<n; j++)
j depends on i.

Approach

eg 1 3 0 0 2 4 8



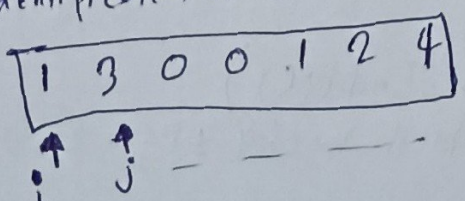


Nearest greater to Right

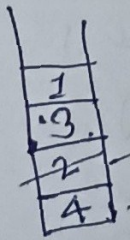
arr[] : 1 3 2 4

o/p : 3 2 4 -1

Identification of stack.



compare kr rhe h.



vector v: -1 4 4 9

s.top() < arr[i] तो pop() कर देंगे

s.top() > arr[i]

vector में push कर देंगे

vector[]: ~~3~~ -1 4 4 9

o/p: 3 4 4 -1

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

~~reverse~~
return v;

three step

vector में push

Stack empty → -1

s.top() > arr[i] → s.top()

s.top() <= arr[i] → pop stack से

unitill i) stack empty

check ii) s.top() greater than arr[i].

code

```
vector<int> v;
stack<int> s;
for(int i= size-1; i>=0; i--)
{ 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(arr[i]);
}
return reverse(v.begin(),
      v.end())
```

Nearest Greater to left.

arr[]: 1 3 2 4

op -1 -1 3 -1

Code

```
vector<int> v;
stack<int> s;
for(int i=0; i<arr.size(); i++)
{  while(s.size()>0 &&
        s.top() <= arr[i])
   {
      s.pop();
   }
   if(s.isEmpty())
      v.push_back(-1);
   else
      v.push_back(arr[i])
}  s.push(arr[i]);
```

Nearest Smaller to left

arr[]: | 4 | 5 | 2 | 10 | 8 |

o/p : -1 4 -1 2 2

Identification

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

| 4 | 5 | 2 | 10 | 8 |

i-1 j

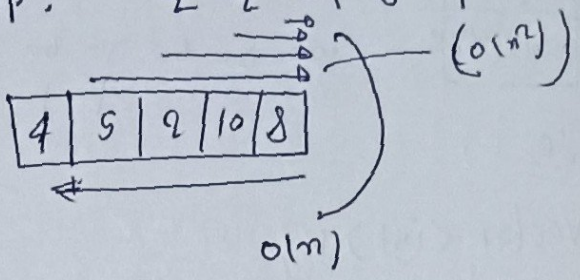
code

```
vector<int> v;
stack<int> s;

for(int i=0; i<n; i++)
{
  while( s.size()>0 && s.top() >= arr[i] ){
    s.pop();
  }
  if(s.isEmpty())
  {
    v.push_back(-1);
  }
  else{
    v.push_back(arr[i]);
  }
  s.push(arr[i]);
}

return v;
```

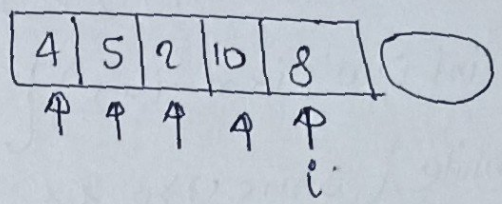
Nearest Smaller to right

arr[]: 4 5 2 10 8

o/p: 2 2 -1 8 -1



Approach



v: | -1 | 8 | -1 | 2 | 2 |

reverse(v) : | 2 | 2 | -1 | 8 | -1 |

6 Stock Span Problem

arr: | 100 | 80 | 60 | 70 | 60 | 75 | 85 |

op: | 1 | 1 | 1 | 2 | 1 | 4 | 6 |

see Consecutive smaller or equal to arr[i].

	0	1	2	3	4	5	6
arr:	100	80	60	70	60	75	85

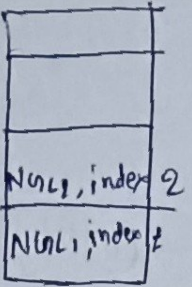
↳ arr in index

v: | -1 | 0 | 1 | 1 | 3 | 1 | 0 |

NGL :

o/p: | 1 | 1 | 1 | 2 | 1 | 4 | 6 |

↳ arr in index − v[i]



100 80 60 70 60 75

4 1

Code

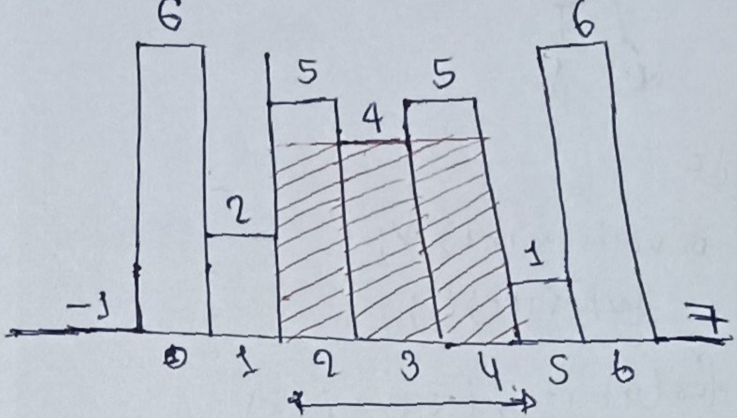
```
vector<int> v;
~~Sack~~
Stack<pair<int,int>> S;
for(int i=0; i<n; i++){
  while(S.size()>0 &&
        S.top().first <= arr[i])
  {
    S.pop().~~first~~;
  }
  if(S.size()==0)
    v.push_back(~~S.top().second~~ -1);
  else
    v.push_back(S.top().second);
  }
  S.push(arr[i], i);
}
for(int i=0; i<v.size(); i++)
{
  v[i] = i - v[i];
}
return v;
```

sum up.

Stack Stuff problem → Greatest smaller → NGL → (NGL index) ↓ | i - NGL |

Maximum Area Histogram

arr[]: 6 2 5 4 5 1 6 ← height



4 × 3 = 12

right

		0	1	2	3	4	5	6
arr[]:		6	2	5	4	5	1	6
NSR	:	1	5	3	5	5	7	7
(Left) NSL	:	-1	-1	1	1	3	-1	5

width[] = 5 - 1 = 4 - 1

= right - left - 1 = 3

arr[]: 6 2 5 4 5 1 6

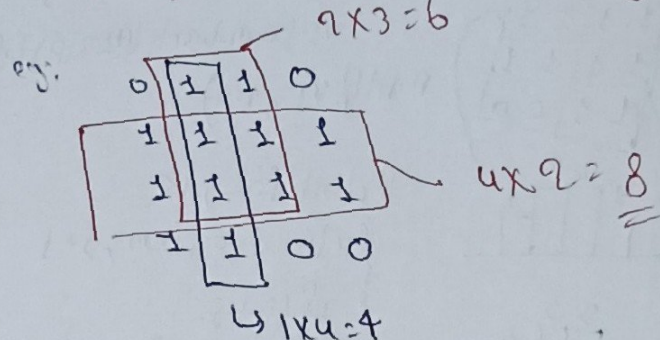
width[]: 1 5 1 3 1 7 1

area[] = arr[] × width[]

area[] = 6 10 5 12 5 7 6

Max Area Rectangle in binary matrix.

e.g.

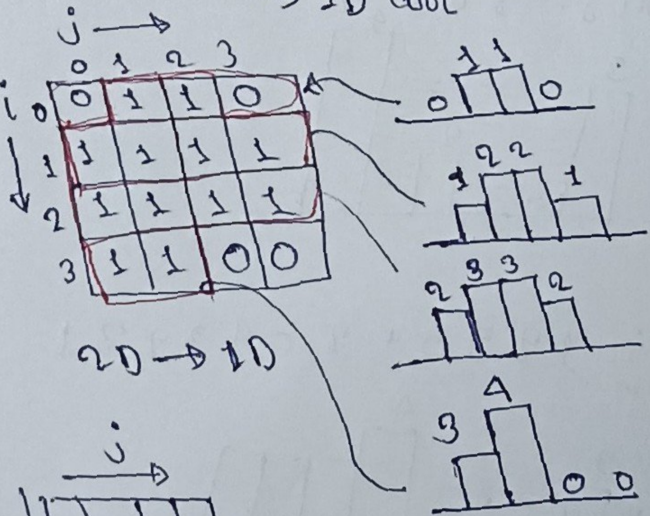



```
MAH(int arr[], int size)
{
    right[] -> NSR(arr, size)      release
    left[] -> NSL(arr, size)
    width[] -> right[i]-left[i]-1
    arr[] -> arr[i] * width[i]
}.  return max in arr[];
```

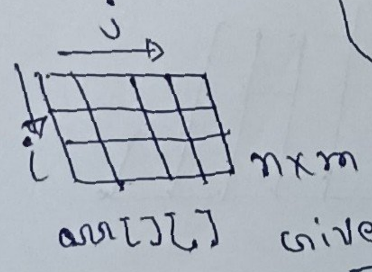
matching

MAH	d/f	given MAH
	1D arr	Natural no.
Q	2D arr.	binary no 0/1

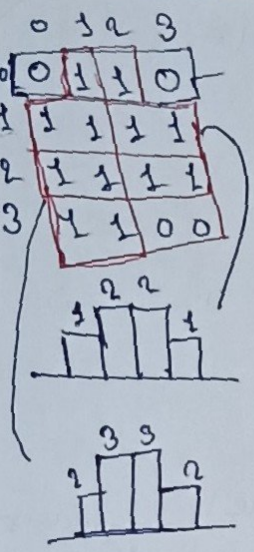
2D arr. → 1D arr

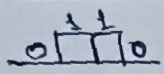


2D → 1D



arr[][] given.



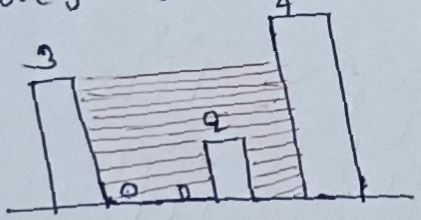


```
for(int j=0; j<m; j++)
    v1.push_back(arr[0][j]
MAH1 = MAH(v1, m)

for(int i=1; i<
  for(int j=0; j<m; j++
  {
    if(int j;
    if(arr[i][j]==0)
      v[j]=0;
    else
      v2[j] = v1[j] + arr[i][j]
  MAH2 = MAH(v2, m)
```

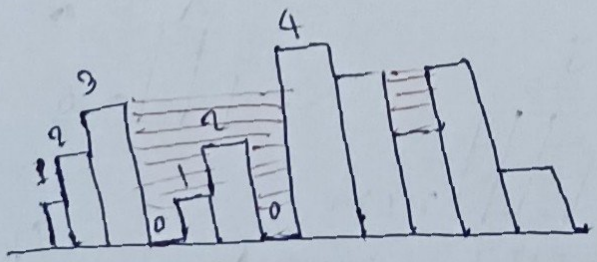
```
for(int j=0; j<m; j++)
{ if(arr[i][j]==0)
   v3[j]=0;
else
   v3[j] = v2[j] + arr[i][j];
}
MAH2 = MAH(v3, m)

return max(MAH1, MAH2, MAH3, MAH4)
```

Rain water Trapping:

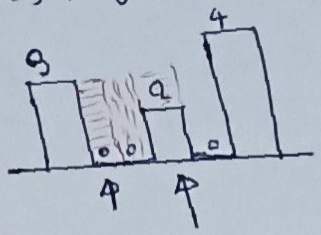
arr[]: 3 0 0 2 0 4



eg2: 1 2 3 0 1 2 0 4 3 2 3 1



arr[]: 3 0 0 2 0 4



min(3,4) - 2 = 1 x 1 = 1

water[i] =

min(maxLeft - maxRight) - arr[i]

arr[]: 3 0 0 2 0 4

maxL[]: 3 3 3 3 3 4

maxR[]: 4 4 4 4 4 4

arr[]: 3 0 0 2 0 4

min(maxL[]: 3 3 3 3 3 4

maxR[]: 4 4 4 4 4 4

minD[] 3 3 3 3 3 4

arr[] 3 0 0 2 0 4

water[] 0 3 3 1 3 0

return sum of water[].

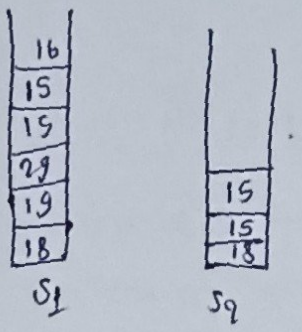
code

```
int maxL = arr[0]
for(int i=1; i<n; i++)
  maxL[i] = max(maxL[i-1],
                maxL[i])
int maxR[n-1] = arr[n-1]
for(int i = n-2; i>=0; i--)
  maxR[i] = max(maxL[i-1],
                maxL[i]);
```

minimum element in stack.
with extra space.

ex 18 19 29 15 15 16

Approach.



Code

```
stack<int> s1;
stack<int> s2;

int getmin()
{ if(s2.size()==0)
    return -1;
  return s2.top();
}

void push(int a)
{ s1.push(a);
  if(s2.size()==0 || a<=s2.top())
    s2.push(a);
  return;
}
int pop(){
  if(s.size()==0)
    return -1;
  int ans = s1.top()
    s1.pop();
  if(ans == s2.top())
    s2.pop();
  return ans;
```