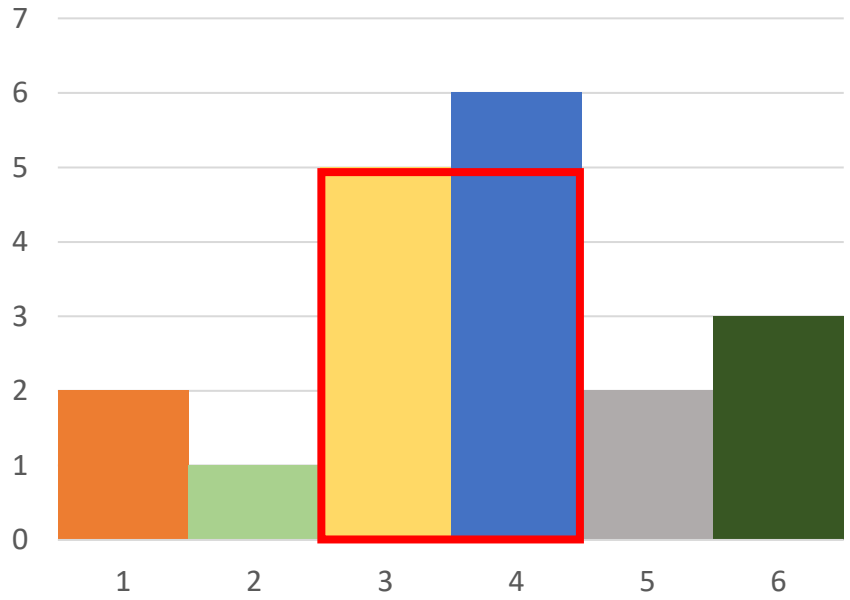
Largest Rectangle in a Histogram



Nihar Ranjan Roy
nihar.ranjan@sharda.ac.in

Given a binary matrix, find the maximum size rectangle binary-sub-matrix with all 1’s.

- Input:

  0 1 1 0

  1 1 1 1

  1 1 1 1

  1 1 0 0

- Output :

Largest rectangular area is 8

Lets convert this problem in to a histogram

- Input:

0 1 1 0

1 1 1 1

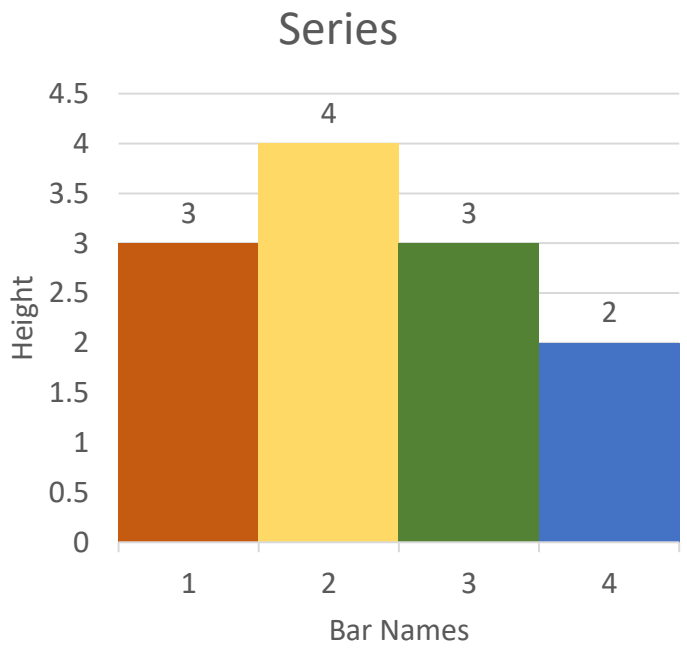
1 1 1 1

1 1 0 0

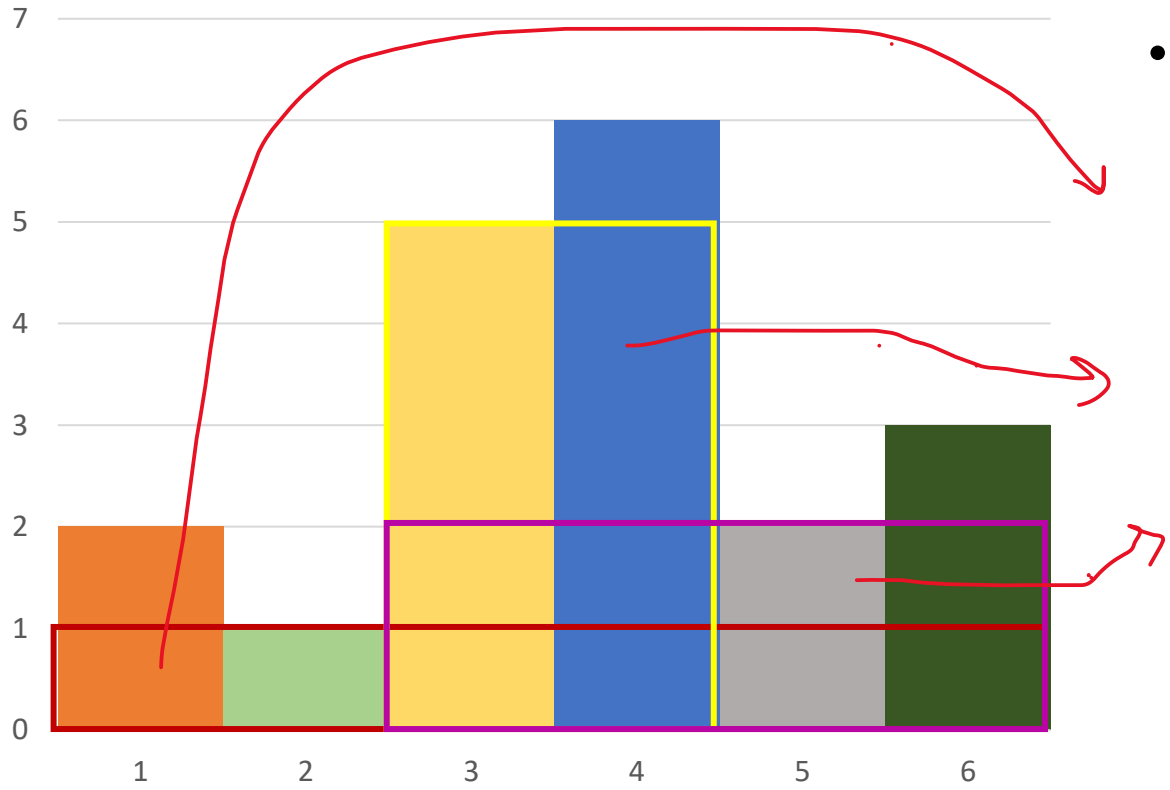
- Output :

Largest rectangular area is 8

```
for i in range(1, len(A)):
            for j in range(len(A[i])):
              # if A[i][j] is 1 then add A[i -1][j]
               if (A[i][j]):
                   A[i][j] += A[i - 1][j]
```



Largest rectangle in a Histogram



- There are many rectangles in this histogram.
- Some of them are

Area=6x1=6

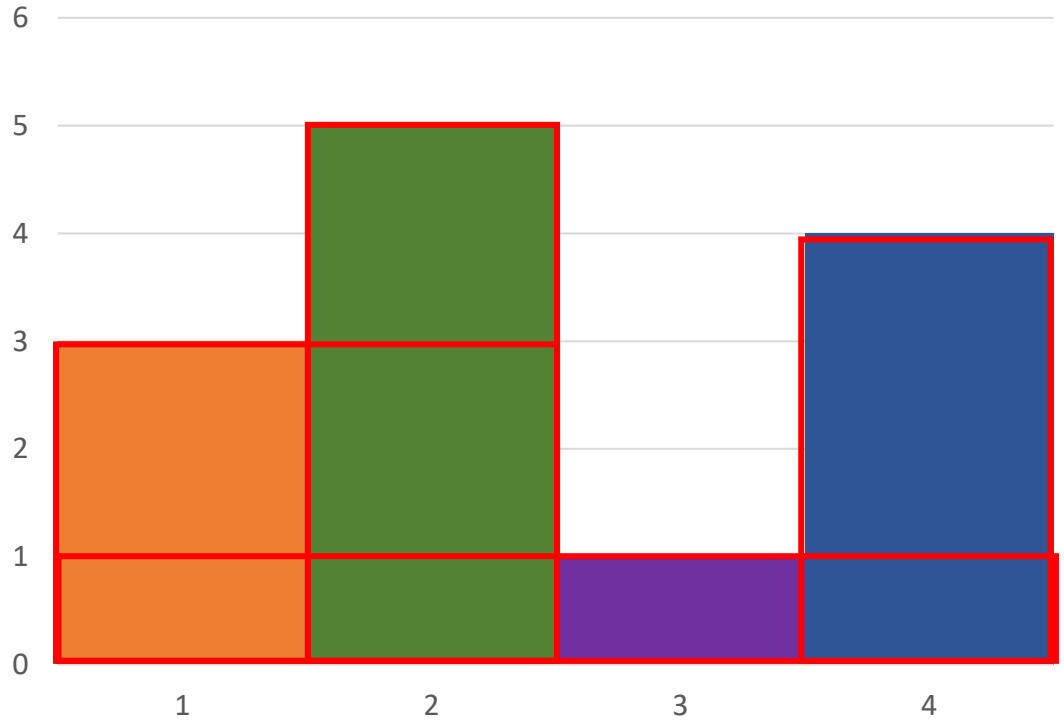
Area=2x5=10

Area=4x2=8

Maximum area=10

- Do you agree that in order to find area of largest rectangle we are going to include at least one bar with its full height?
- Lets try to include one by one all the bars and try to find the max rectangular area

Bar included in full	Area of the largest rectangle (height X bars)	Max Area
1	3x2=6	6
2	5x1=5	
3	1x4=4	
4	4x1=4	



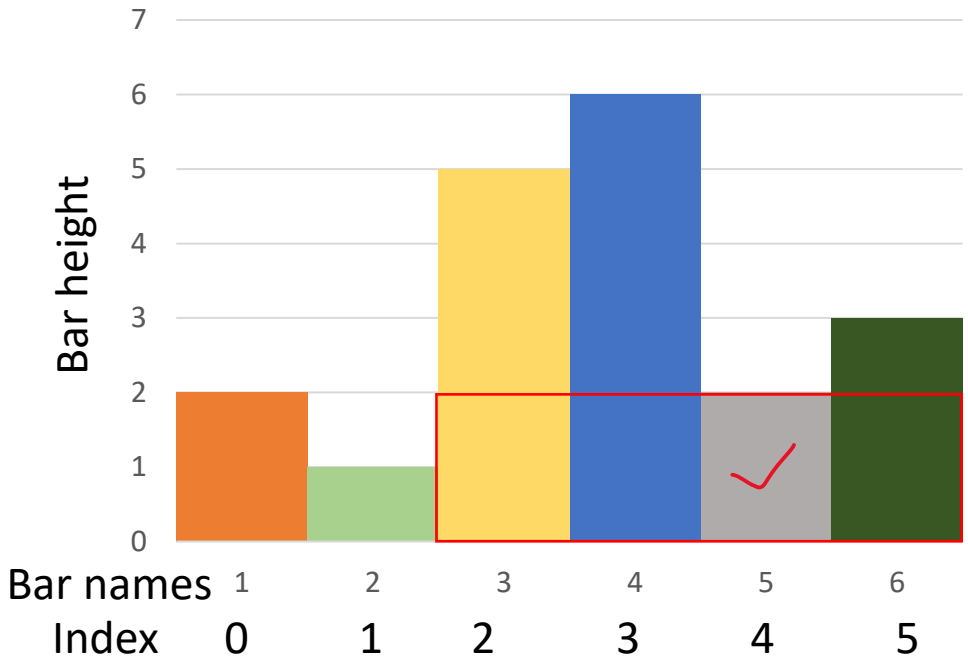
Points to note:

1. One by one find the max rectangular area by including each bar
2. Find the maximum of (1) calculated for each bar

Finding maximum area for a bar Solution No-2

Rule:

1. Find the nearest left bar with height< current bar
2. Find the nearest right bar with the height< current bar



- Find the largest rectangular area for bar 5
- Travel left and stop just before the nearest left bar with height< current bar (bar 5). In this case it is bar no 2,so left index for bar 5 is 2
- Travel right and stop just before the nearest right bar with height< current bar(bar 5). In this case it is bar 6, therefore right index for bar 5 is 5.
- Area=(right index-left index+1)x height of current bar

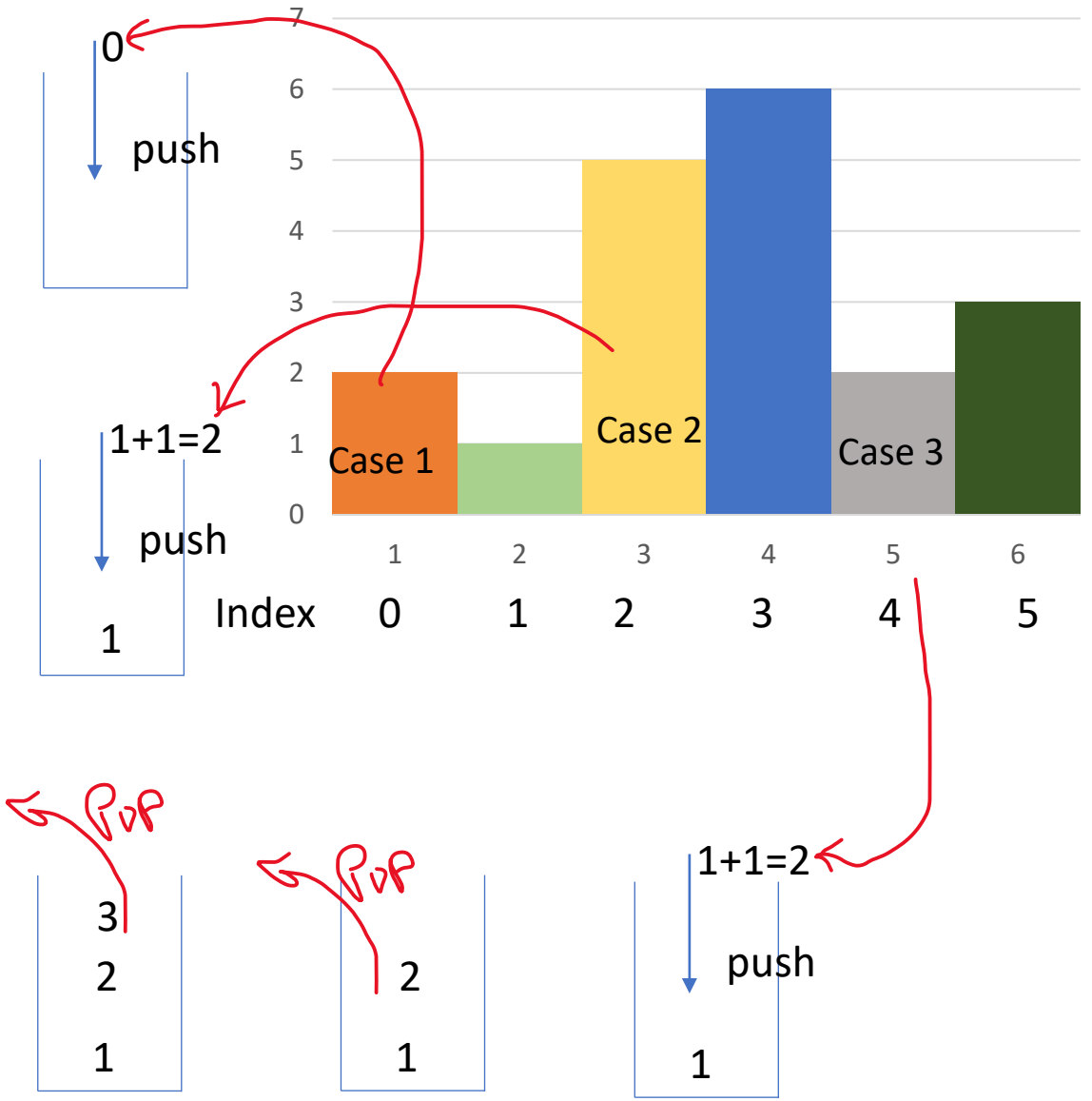
  =(5-2+1)*2=8

Complexity of this approach?? (BRUTE FORCE)

Since we have to traverse the entire list of bars to find left and right limits for every bar. Thus if we have N bars then its complexity would be $O(N^2)$

Solution No-3

- Keep track of previously visited smaller bars and maintain its list in a stack
- The previously visited bars must be in increasing order of their height in the stack. That is the bar at the top must the with tallest bar.
- Case 1: if the stack is empty then left index for current bar would be 0 and we push the index of the current bar to the stack
- Case 2: If top of the stack refers to a bar with height less than the current bar then. The left limit for current bar would be top value of stack+1 and we push the current bars index to the stack
- Case 3: If top of the stack refers to a bar with height greater than the current bar then we pop this value and recheck it unless we get a case similar to case 1.

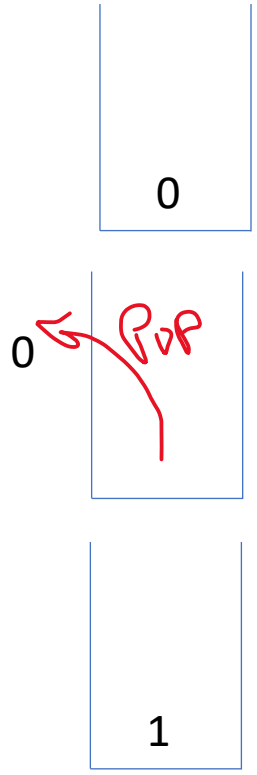


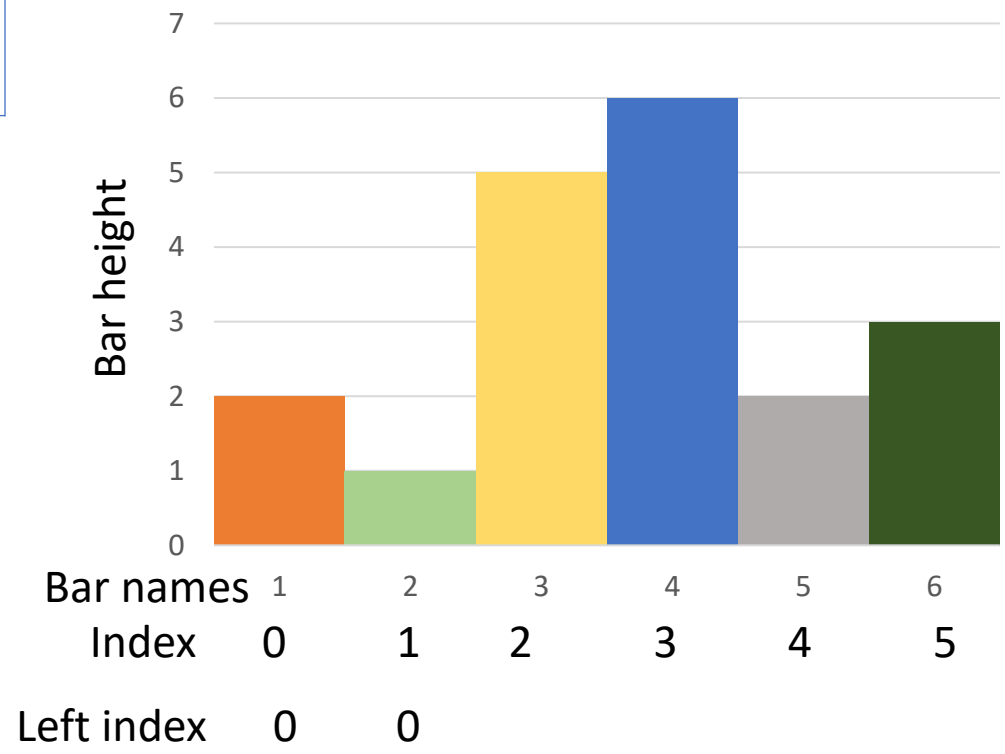
Find the left limits of every bar

- Initially stack is empty
- **Left limit of bar 1 with height 2**
- Since there is no bar in the stack, the left limit is the extreme left bar here it is this bar itself ie index 0. So push 0 to the stack
- **Left limit of bar 2, with height 1**
- The top of the stack has index 0 which corresponds to bar 1. Bar 1 has height > current bar. So we will pop out this value.
- Recheck the stack for values now stack is empty. This means till extreme left we dnot have any bar with height less than current bar so we put its left limit as 0 and push 1 (index of current bar) in the stack

1

1



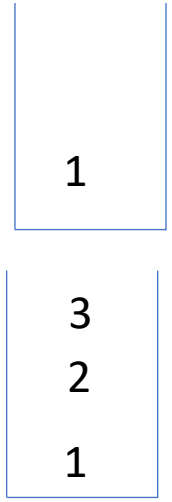


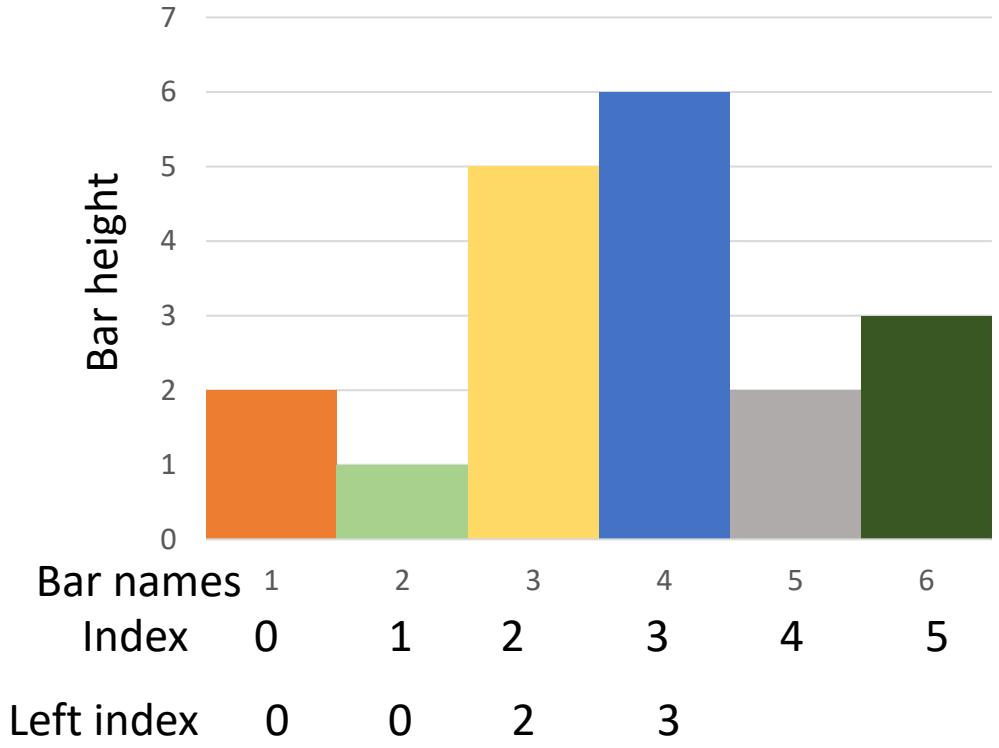
Find the left limits of every bar

- **Left limit of bar 3 with height 5**
- The top of the stack has bar no 1 which is less than height of current bar. So left limit for bar three would be its current index i.e top of stack+1 and we will push 2 into the stack.
- **Left limit of bar 4, with height6**
- The top of the stack has index 2 which corresponds to bar 3. Bar 3 has height < current bar. So left limit for this bar would be its own index and we push its index to the stack ie 3

5

6





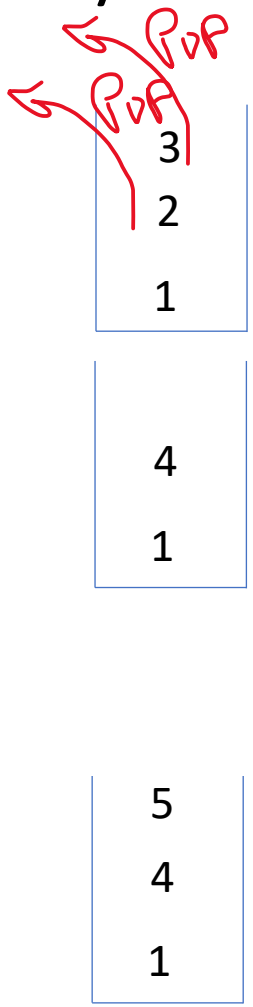
Find the left limits of every bar

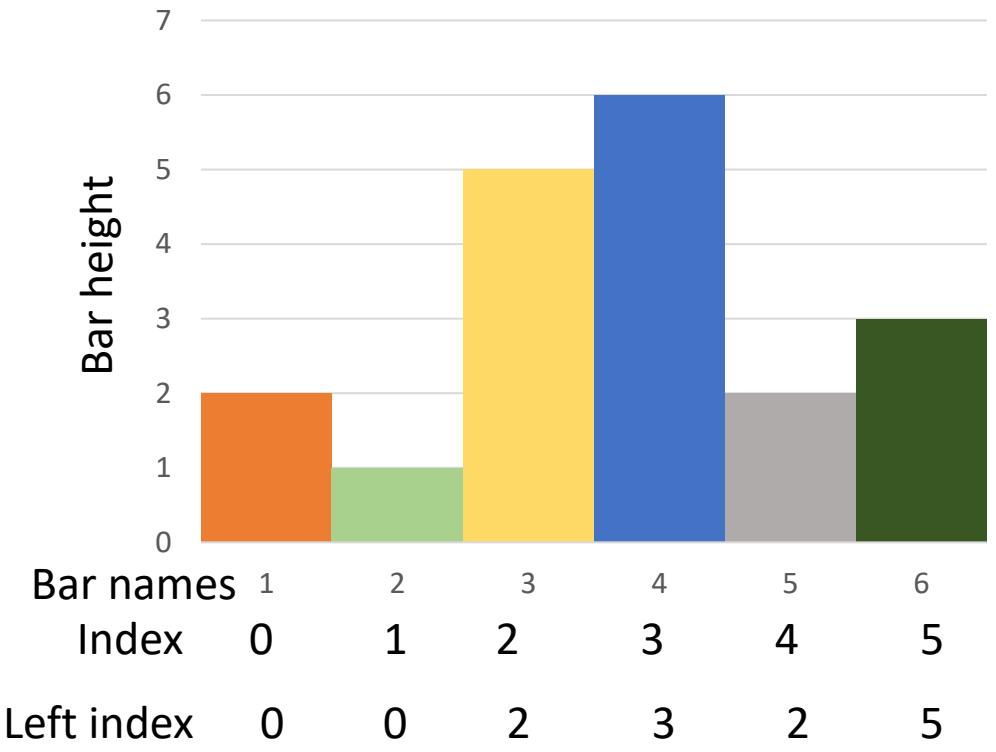
5

- **Left limit of bar 5 with height 2**
- The top of the stack has bar 4 whose height is > current bar so pop it out.
- Next, the top of the stack refers to bar 3, whose height > current bar, so pop it out.
- Next, the top of the stack refers to bar 2 whose height is < current bar. So we put index of just the previous bar as left index and push the current bar's index to the stack i.e. 4.

6

- **Left limit of bar 6, with height 3**
- The top of the stack has index 4 which corresponds to bar 5. Bar 5 has height < current bar. So left limit for this bar would be its own index (5) (in another way top of the stack+1) and we push its index to the stack i.e. 5



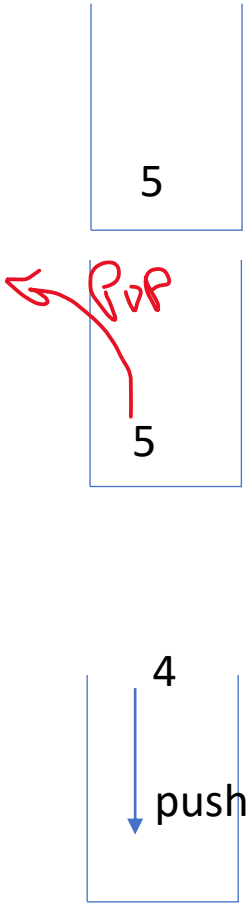


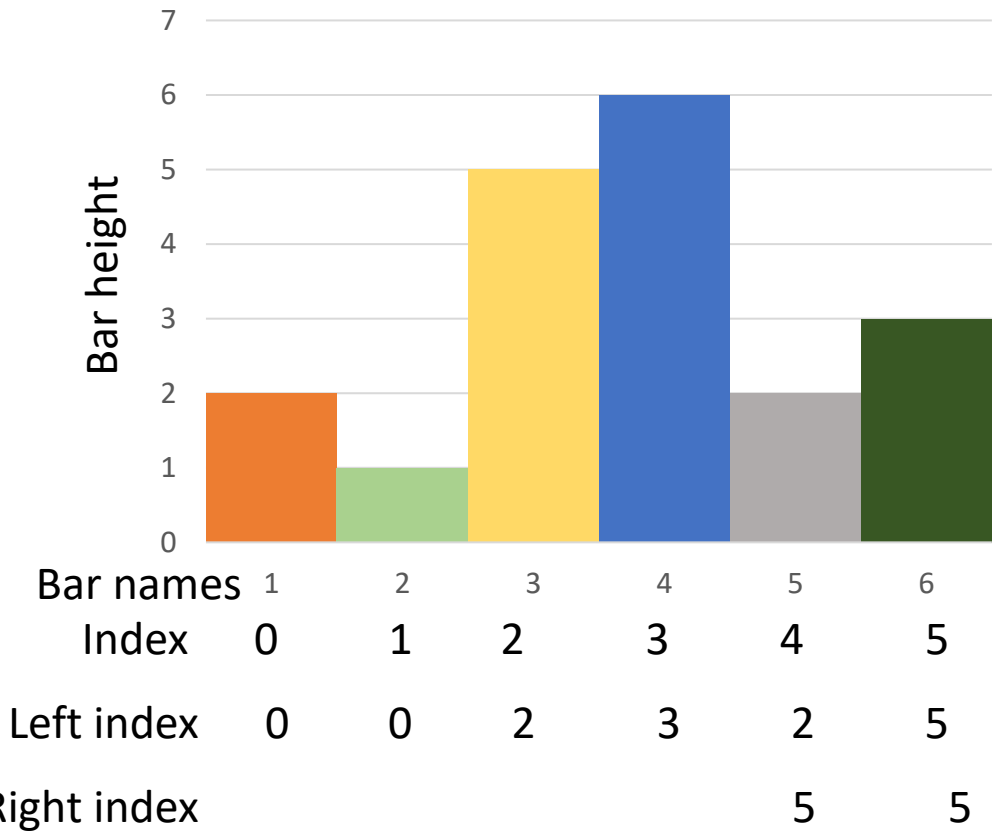
Find the RIGHT limits of every bar

6

5

- **right limit of bar 6 with height 3**
- Since stack is empty then the right limit for current bar is the extreme right index or number of bars-1=5 in this case. We push current bar's index in the stack ie 5
- **Left limit of bar 5, with height 2**
- The top of the stack has index 5 which corresponds to bar 6. Bar 6 has height > current bar. So we pop it out. Next when we read it stack is empty. So right limit for this bar is the extreme right index or number of bars-1=5 in this case. We push current bar's index in the stack ie 4





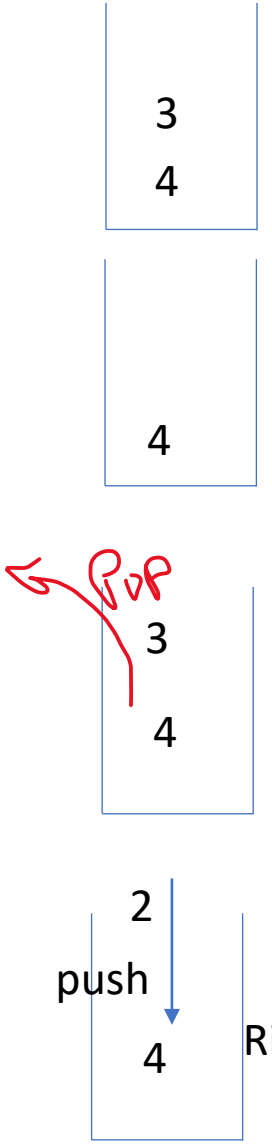
Find the RIGHT limits of every bar

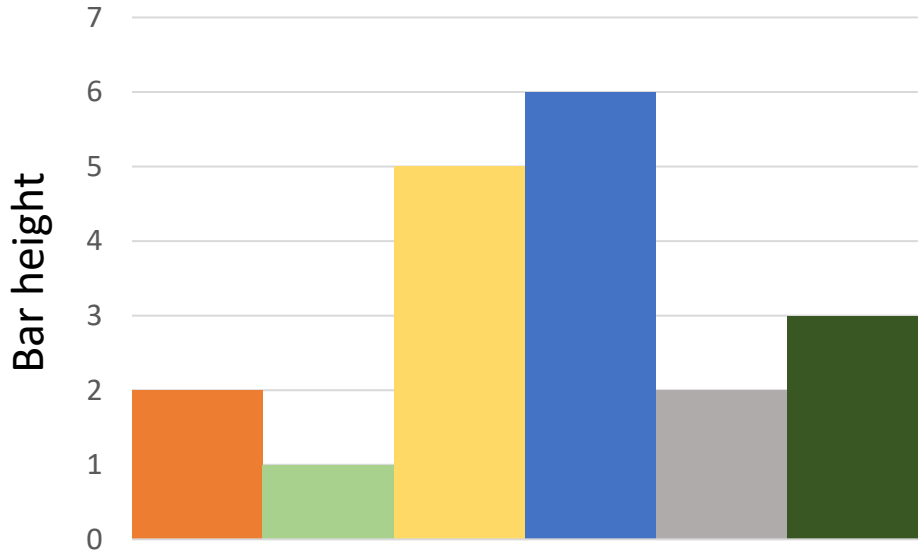
4

- **right limit of bar 4 with height 6**
- Since the top of the stack refers to bar 4 whose height is less than current bar. So right limit for current bar is top of stack-1 i.e 4-1=3 and we push current index ie 3 to stack

6

- **right limit of bar 3, with height 5**
- The top of the stack has index 3 which corresponds to bar 4 with height 6. Bar 4 has height > current bar. So we pop it out.
- Next top ie index 4 referes to bar 5 which is <current bar's height. So right limit for current bar is top-1 ie 4-1=3 and we push current bars index ie 2 to stack





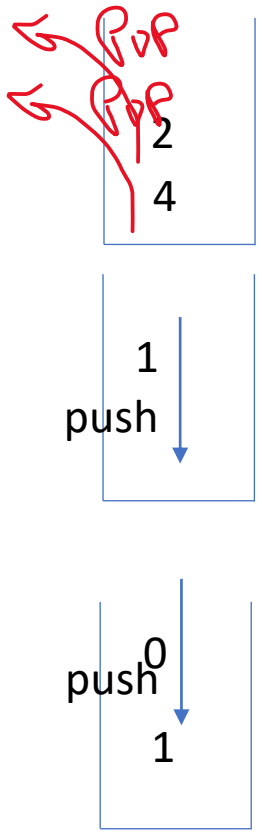
Index	0	1	2	3	4	5
Left index	0	0	2	3	2	5
Right index			3	3	5	5

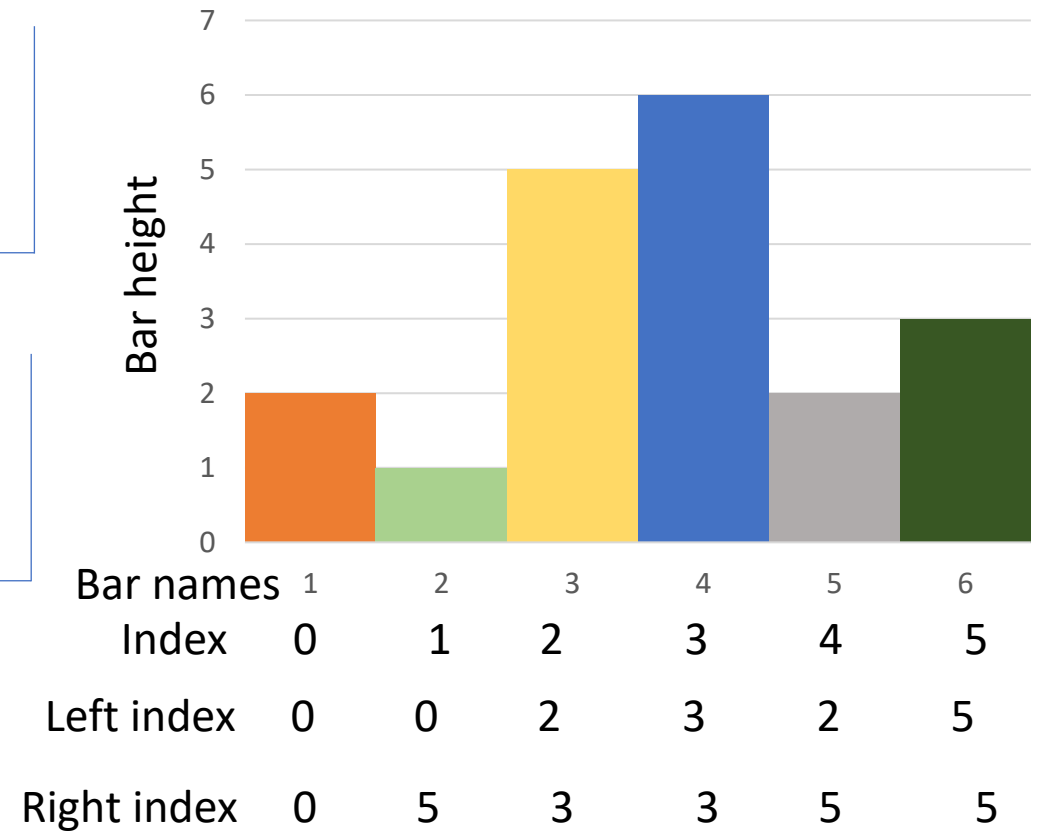
Find the RIGHT limits of every bar

2

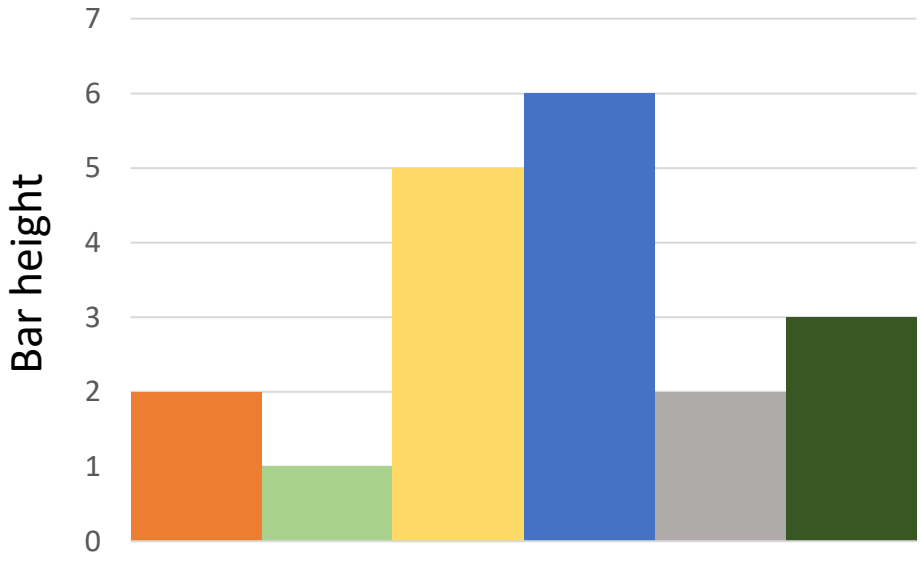
6

- **right limit of bar 2 with height 1**
- Since the top of the stack refers to bar 3 whose height is >current bar. So we pop it. Next top refers to bar 5, whose height > current bars height so we pop it.
- Now the stack is empty so current bars right index is 5 and we push current bars index ie 1 to stack
- **Left limit of bar 1, with height 2**
- The top of the stack has index 1 which corresponds to bar 2 with height 1. Bar 1 has height < current bar. So current bar's right limit is top index-1 ie 1-1=0 and we push current bars index (0) to stack





Final Calculation



Bar no	height	Left limit	Right limit	Area=(right-left+1)x height
1	2	0	0	2
2	1	0	5	6
3	5	2	3	10
4	6	3	3	6
5	2	2	5	8
6	3	5	5	3

Maximum rectangle =max(2,6,10,6,8,3)=10

Complexity of this algorithm= O(N)

- Once we travel from left to right to find left limit
- Once we travel from right to left to find right limit
- Once we travel to calculate area.

```
def maxArea(bars):
        tempstack = []
        leftlimit=[]
        rightlimit=[]
        i=0;
        while (i < len(bars)):
                while(True):
                        if (len(tempstack) == 0):
                                tempstack.append(i)
                                leftlimit.append(0)
                                i=i+1
                                break
                        elif bars[tempstack[-1]] < bars[i]:
                                leftlimit.append(tempstack[-1]+1)
                                tempstack.append(i)
                                i=i+1
                                break
                        else:
                        tempstack.pop();
    print("Left Limits",leftlimit)


print("Maximum Area",maxArea([2,1,5,6,2,3]))
```

```
tempstack = []
i=len(bars)-1
while (i >=0):
       while(True):
            if (len(tempstack) == 0):
                tempstack.append(i)
                rightlimit.append(len(bars)-1)
                i=i-1
                break
            elif bars[tempstack[-1]] < bars[i]:
                rightlimit.append(tempstack[-1]-1)
                tempstack.append(i)
                i=i-1
                break
            else:
                tempstack.pop();
    rightlimit=rightlimit[::-1]
    print(rightlimit)
```