HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Applied Algorithm Lab

Largest black sub-rectangle

ONE LOVE. ONE FUTURE.

Largest black sub-rectangle

- Given a rectangle with cell in black or white. Find the largest sub-rectangle with all black cell.
- The rectangle is represented by a 0-1 NxM matrix:

  A[i,j] = 1 represents cell (i,j) as a black cell, and

  A[i,j] = 0 represents cell (i,j) as a white cell
- Output: the area of the sub-rectangle

Largest black sub-rectangle

- Example

stdin	stdout

4 4
0 1 1 1
1 1 1 0
1 1 0 0
1 1 1 0
	6

Largest black sub-rectangle

- Idea to solve:
  - Travel the rows
  - At each row: find the largest subrectangle end at that row

Largest black sub-rectangle

- At each row: find the largest sub-rectangle end at that row
  - To get information for row i: use the histogram instead of full matrix
  - example: only need [2,3,0,0]

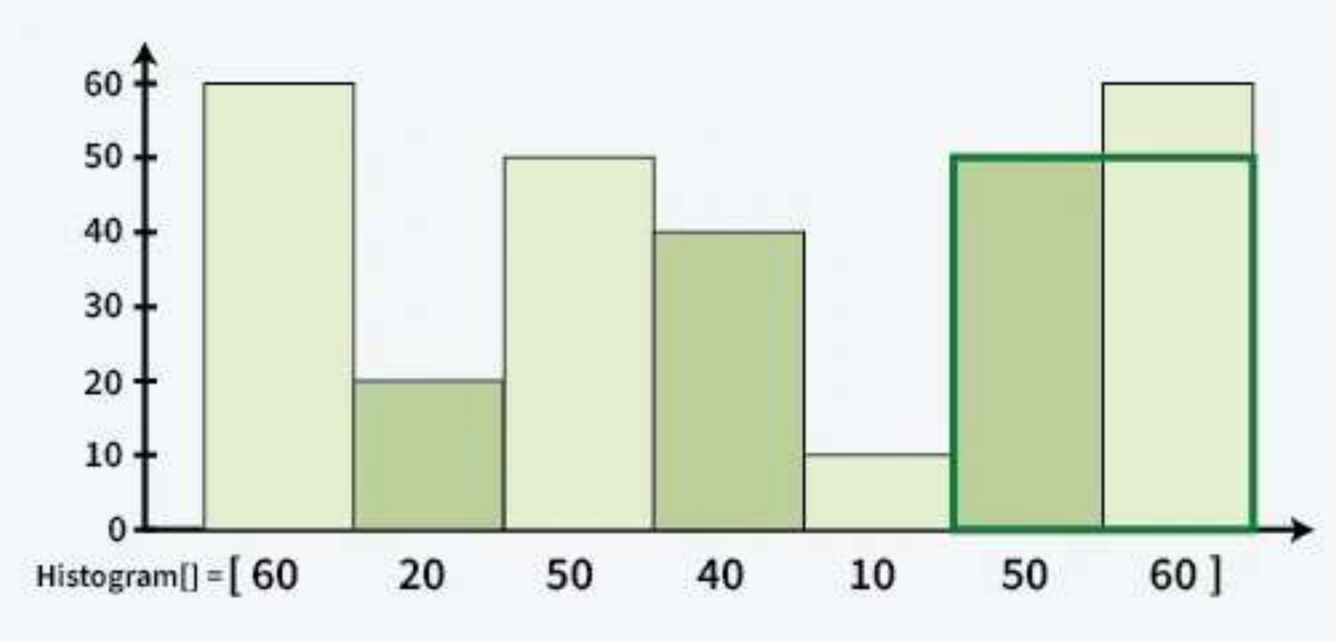
    to find the rectangle for row 3

0	1	1	1
1	1	1	0
1	1	0	0
1	1	1	0

0	1	1	1
1	2	2	0
2	3	0	0
3	4	1	0

Largest black sub-rectangle

- At each row: find the largest sub-rectangle end at that row
  - expand each column
    to find the rectangle with
    that column's height
    - col1: width=1 -> S = 60
    - col2: width=4 -> S = 80
    - col3: width=1 -> S = 50
    - col4: width=2 -> S = 80
    - col5: width=7 -> S = 70
    - col6: width=2 -> S = 100
    - col7: width=1 -> S = 60

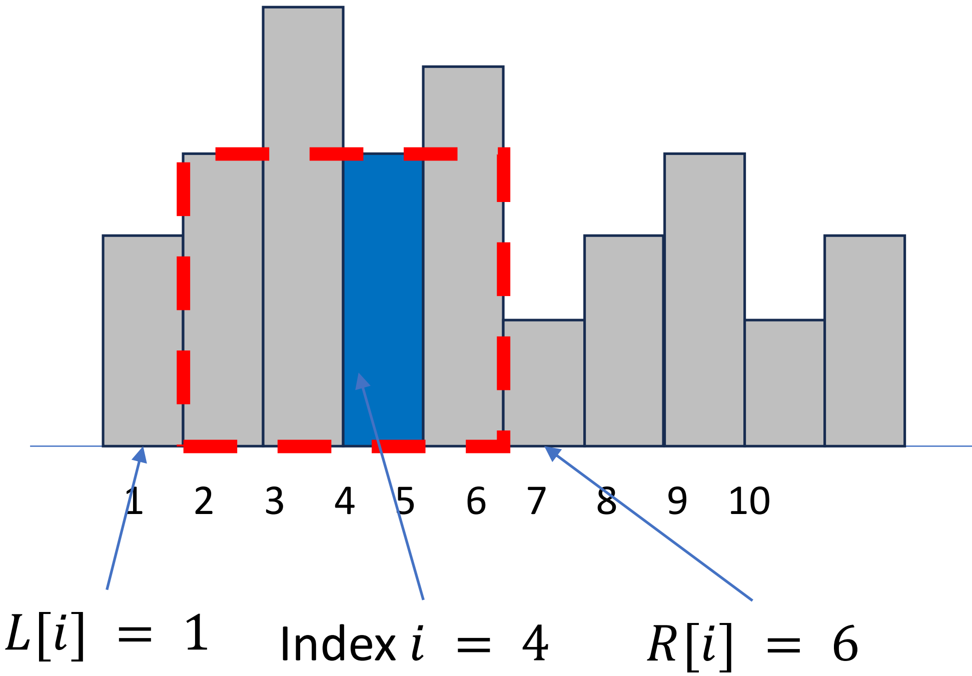


- programming...

Largest black sub-rectangle

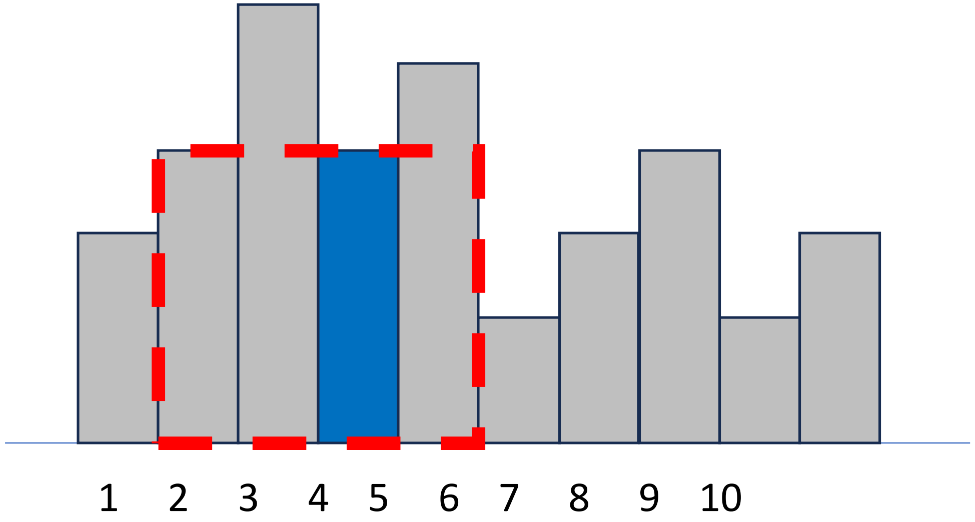
- Solving histogram problem
- Cut out a rectangle from a given histogram shape such that the area is maximal
- Input: a sequence of m column, the (non-negative) height of column i is $h[i], i = 1,2, \ldots, m$
- Border: $h[0] = -1, h[m+1] = -1$
- Data structures: for each column i, $i = 1, \ldots m$
  - $R[i]$: the smallest index j $(i < j)$: $h[i] > h[j]$
  - $L[i]$: the highest index j $(j < i)$: $h[i] > h[j]$
- Area of the largest rectangle obtained by expanding (both left and right directions) from column i is:

$$h[i] * (R[i] - L[i] - 1)$$



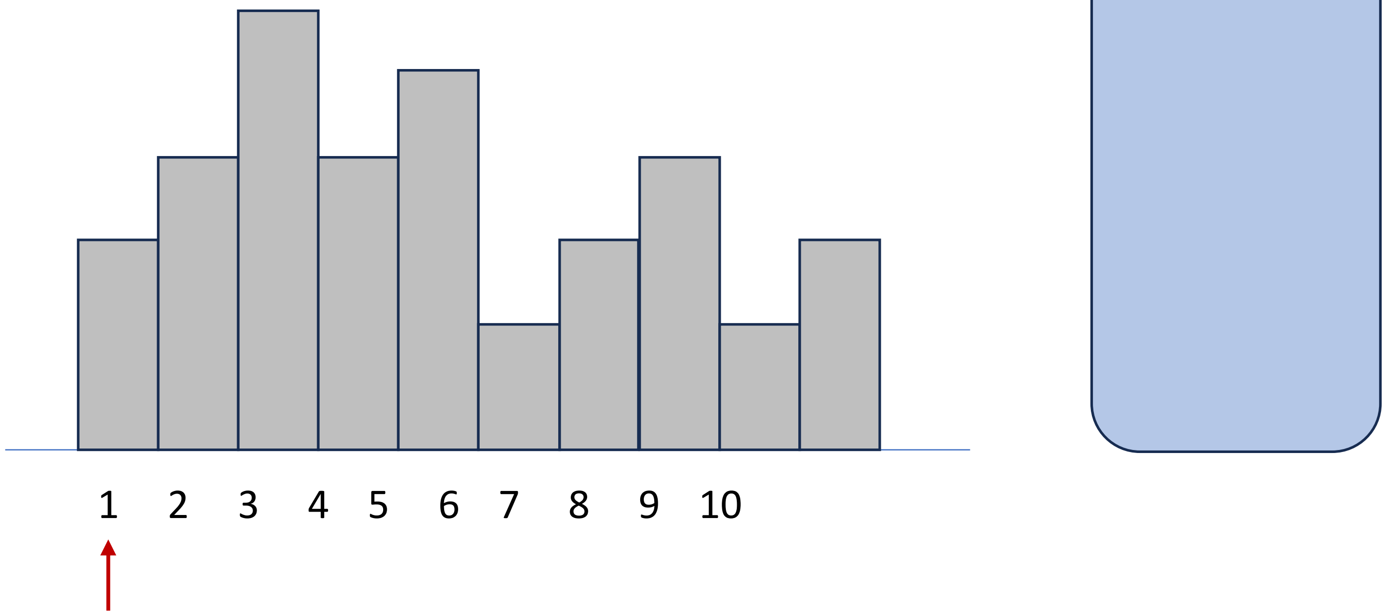
Method

- ➢ Solving histogram problem
- Compute $R[i]$:
  - Explore columns from left to right, maintain a stack S containing indices i of columns waiting for the computation of $R[i]$
  - For each column j:
    - while $h[j] < h[S.top]$ do
      - pop an index i out of S $(i = S.top)$
      - assign $R[i] = j$
    - Push j into S



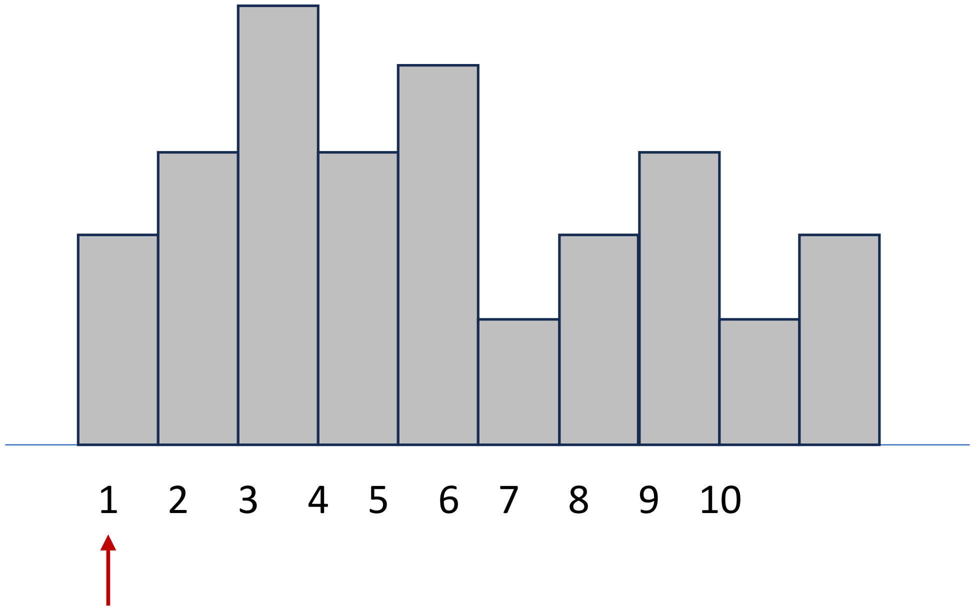
Method

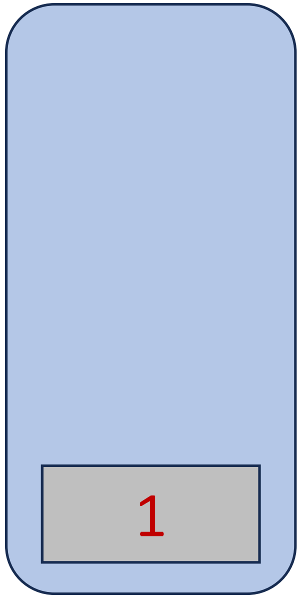
- ➢ Solving histogram problem
- Compute $R[i]$:
    - For each column j:
        - while $h[j] < h[S.top]$ do
            - pop an index i out of S $(i = S.top)$
            - assign $R[i] = j$
        - Push j into S



Method

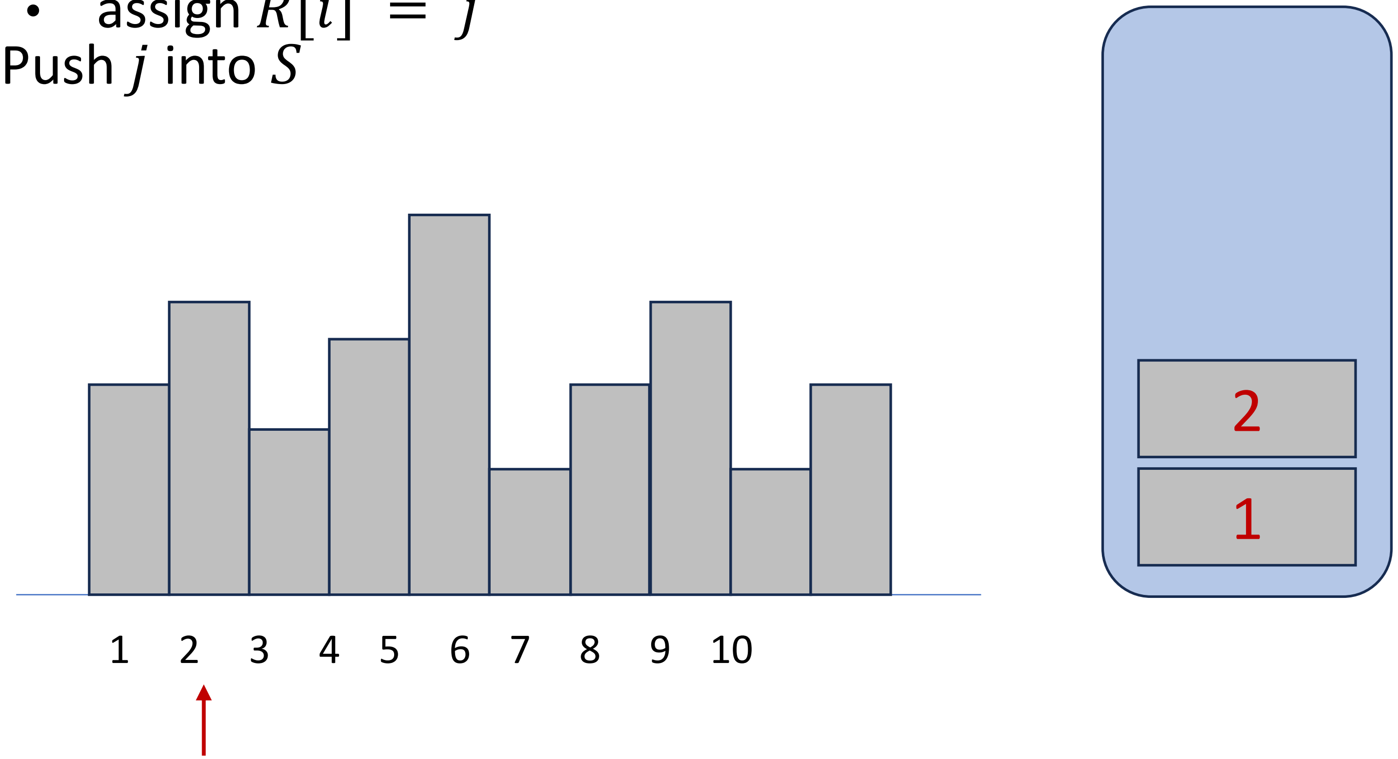
- Solving histogram problem
- Compute $R[i]$:
  - For each column j:
    - while $h[j] < h[S.top]$ do
      - pop an index i out of S $(i = S.top)$
      - assign $R[i] = j$
    - Push j into S





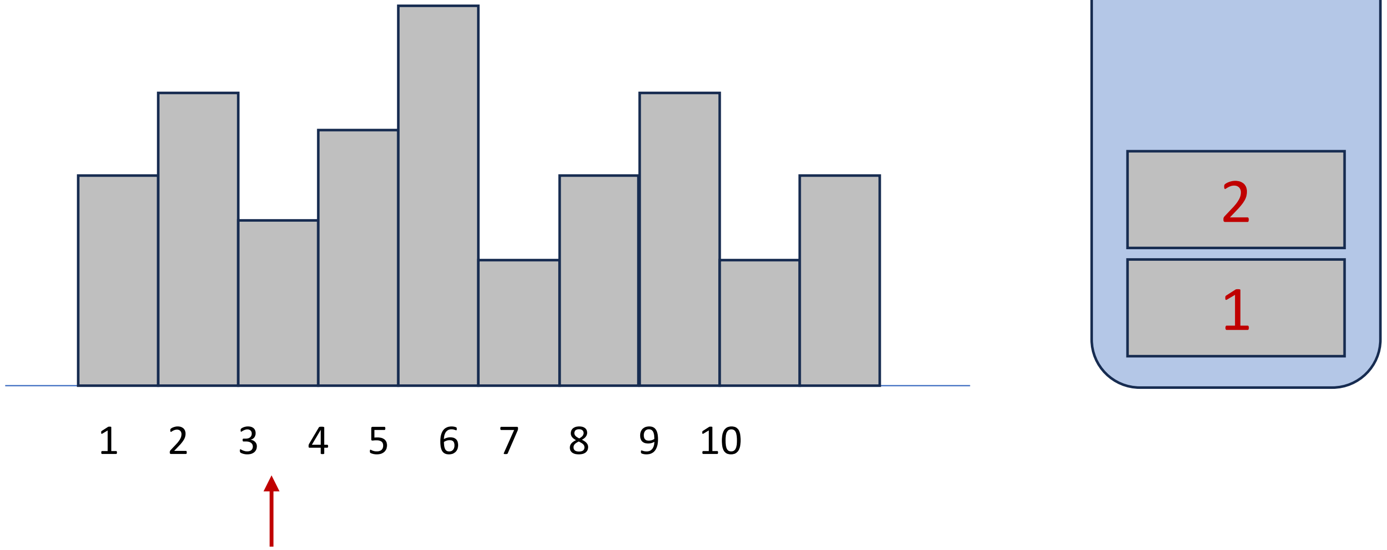
Method

- ➢ Solving histogram problem
- Compute $R[i]$:
  - For each column j:
    - while $h[j] < h[S.top]$ do
      - pop an index i out of S $(i = S.top)$
      - assign $R[i] = j$
    - Push j into S



Method

- Solving histogram problem
- Compute $R[i]$:
    - For each column j:
        - while $h[j] < h[S.top]$ do
            - pop an index i out of S $(i = S.top)$
            - assign $R[i] = j$
        - Push j into S

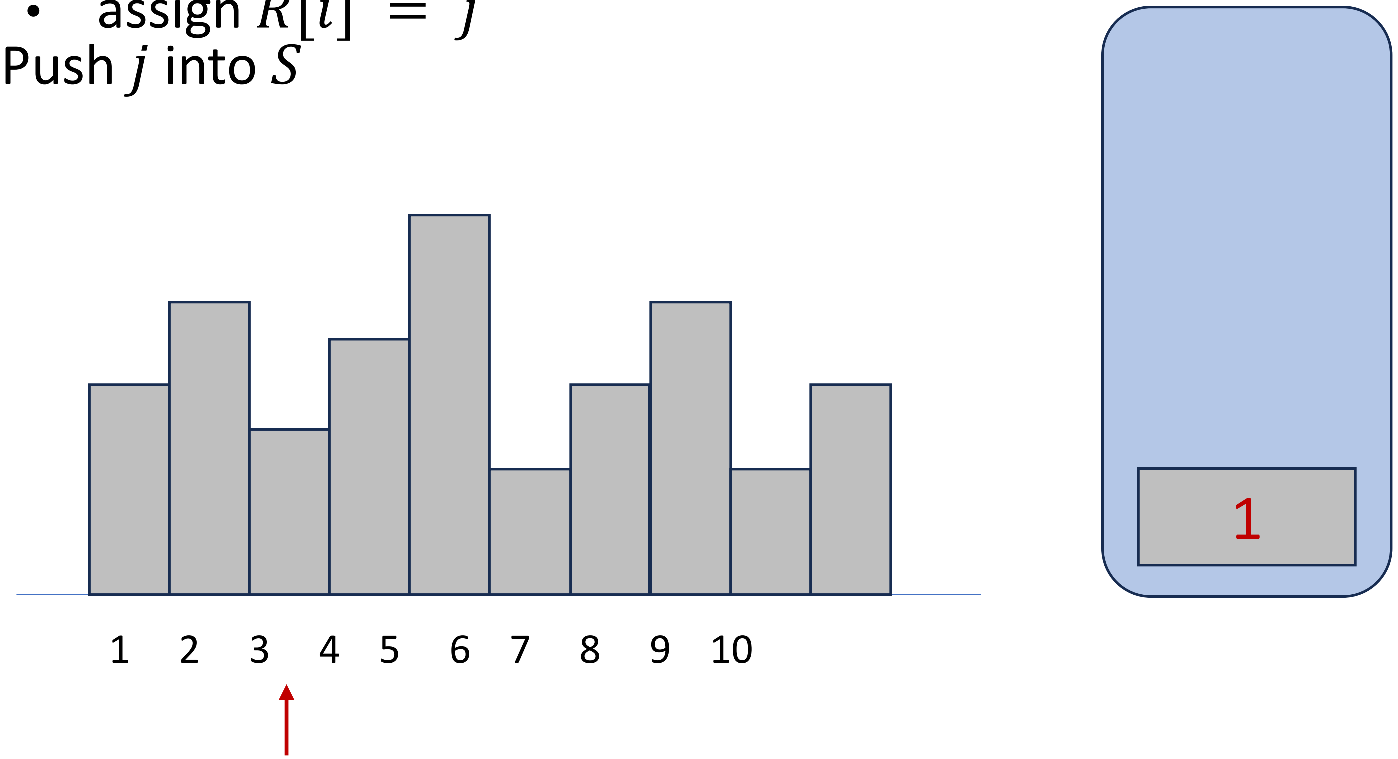


Method

➢ Solving histogram problem

- Compute $R[i]$:
  - For each column j:
    - while $h[j] \ < \ h[S.top]$ do
      - pop an index i out of S $(i \ = \ S.top)$
      - assign $R[i] \ = \ j$
    - Push j into S

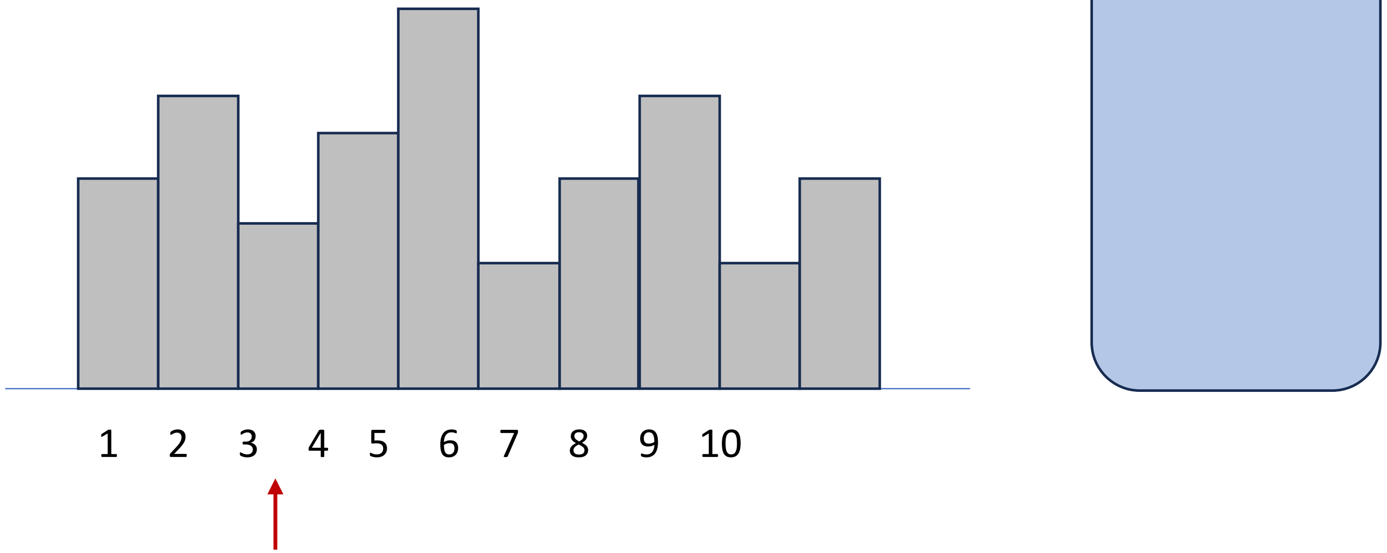
$$R[2] = 3$$



Method

➢ Solving histogram problem

```
• Compute $R[i]$:
    • For each column $j$:
        • while $h[j] < h[S.top]$ do
            • pop an index $i$ out of $S$ ($i = S.top$)
            • assign $R[i] = j$
        • Push $j$ into $S$
```



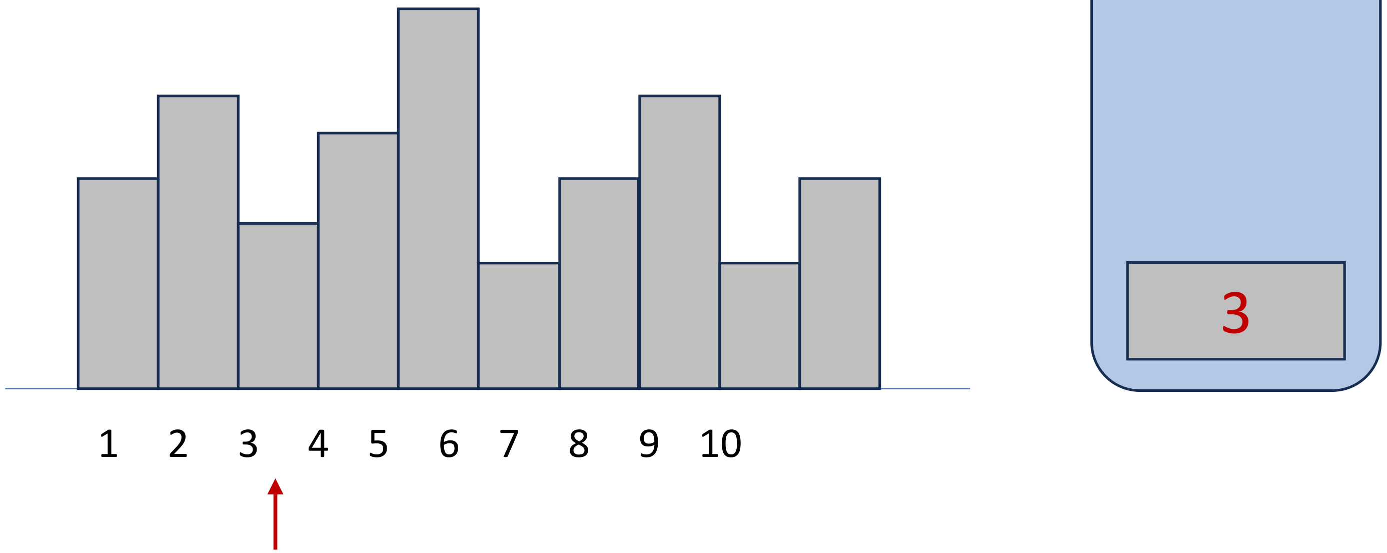
$R[2] = 3$

$R[1] = 3$

Method

- Solving histogram problem
- Compute $R[i]$:

```
For each column $j$:
    while $h[j] < h[S.top]$ do
        pop an index $i$ out of $S$ ($i = S.top$)
        assign $R[i] = j$
    Push $j$ into $S$
```



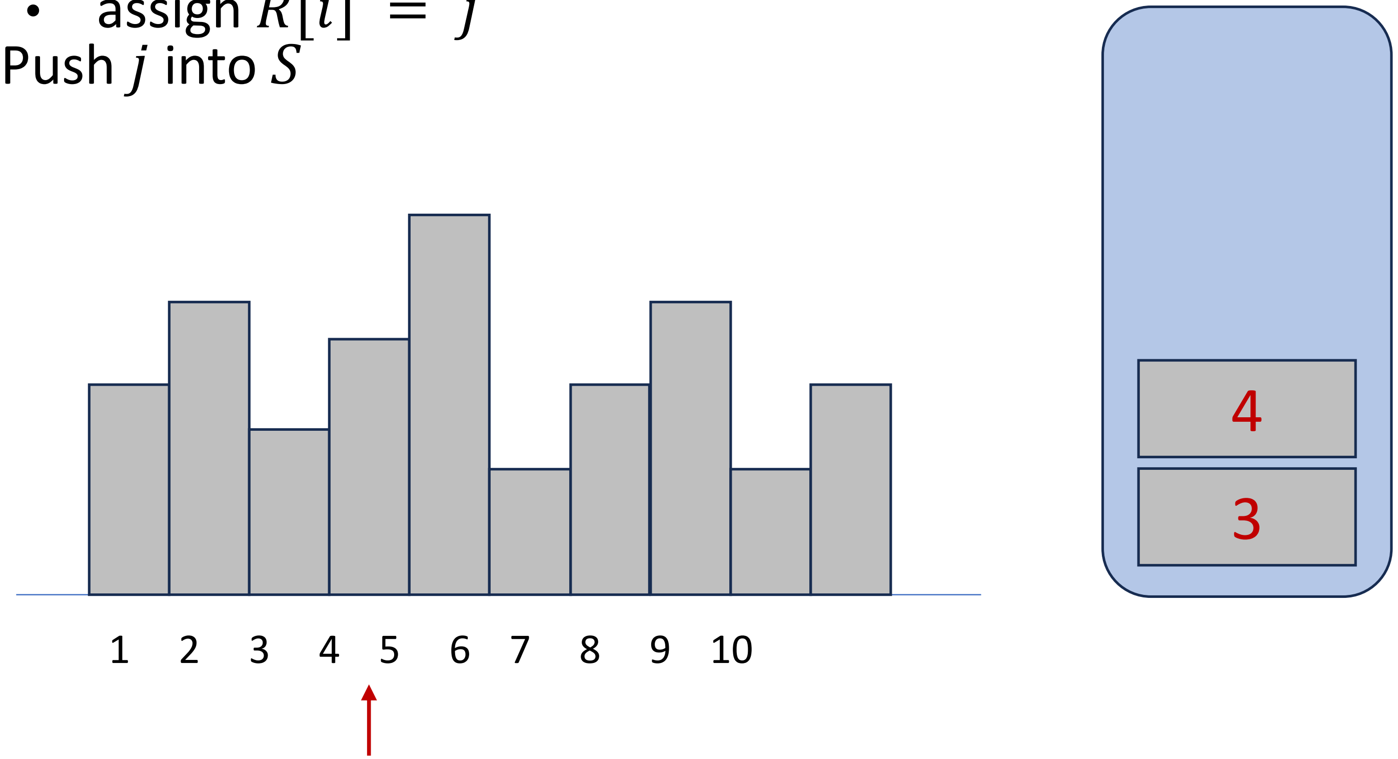
$R[2] = 3$

$R[1] = 3$

Method

- ➢ Solving histogram problem

```
• Compute $R[i]$:
    • For each column $j$:
        • while $h[j] < h[S.top]$ do
            • pop an index $i$ out of $S$ ($i = S.top$)
            • assign $R[i] = j$
        • Push $j$ into $S$
```



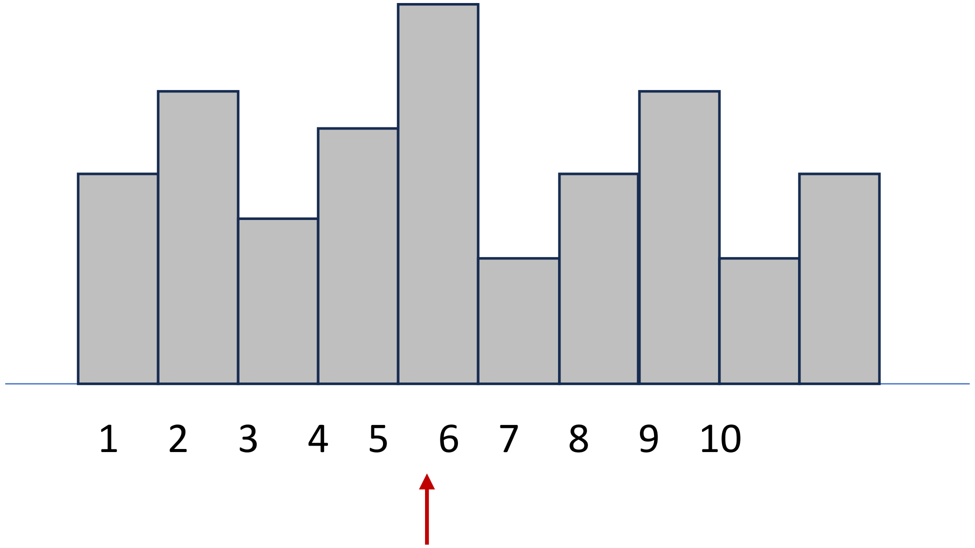
$R[2] = 3$

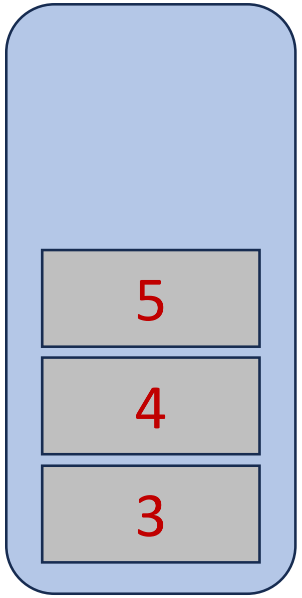
$R[1] = 3$

Method

- Solving histogram problem
- Compute $R[i]$:

```
For each column $j$:
    while $h[j] < h[S.top]$ do
        pop an index $i$ out of $S$ $(i = S.top)$
        assign $R[i] = j$
    Push $j$ into $S$
```





$R[2] = 3$

$R[1] = 3$

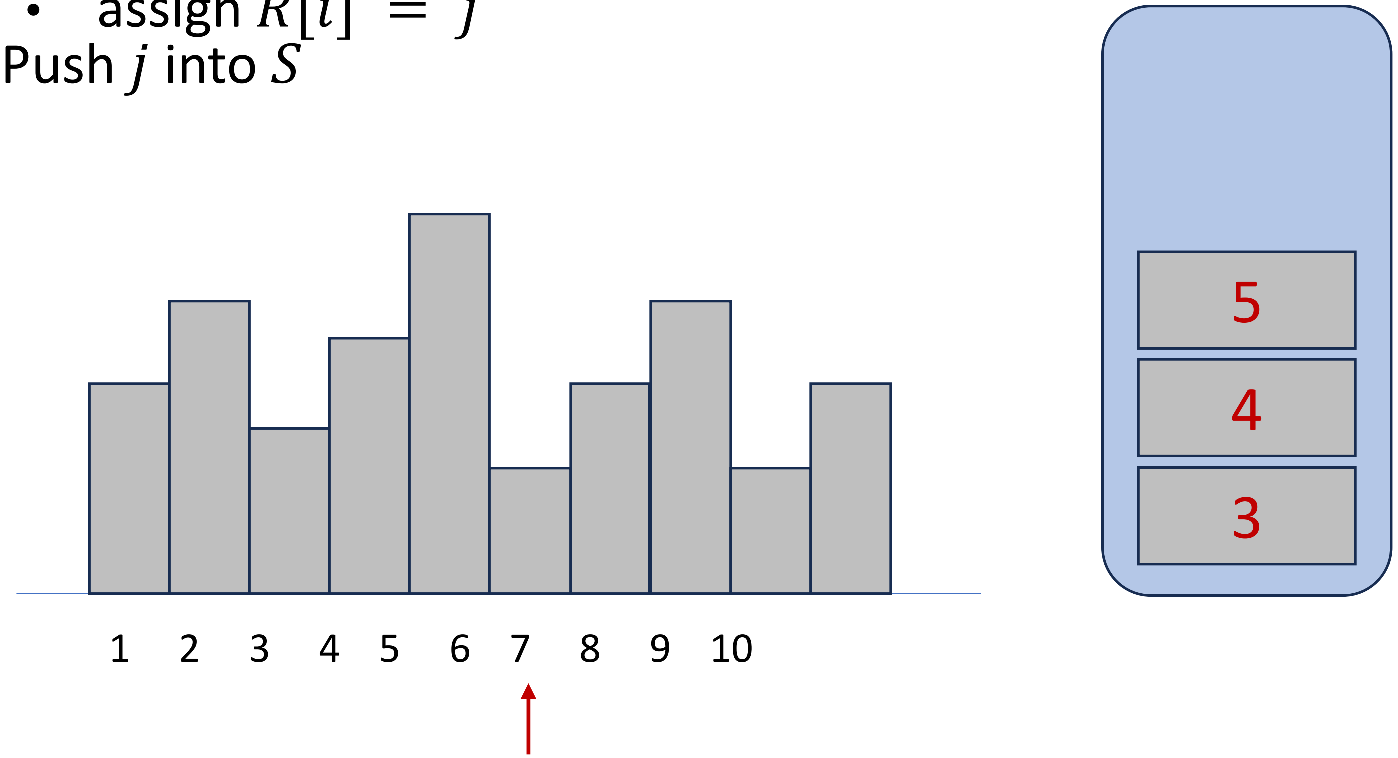
Method

- Solving histogram problem

```
• Compute $R[i]$:
    • For each column $j$:
        • while $h[j] < h[S.top]$ do
            • pop an index $i$ out of $S$ $(i = S.top)$
            • assign $R[i] = j$
        • Push $j$ into $S$
```

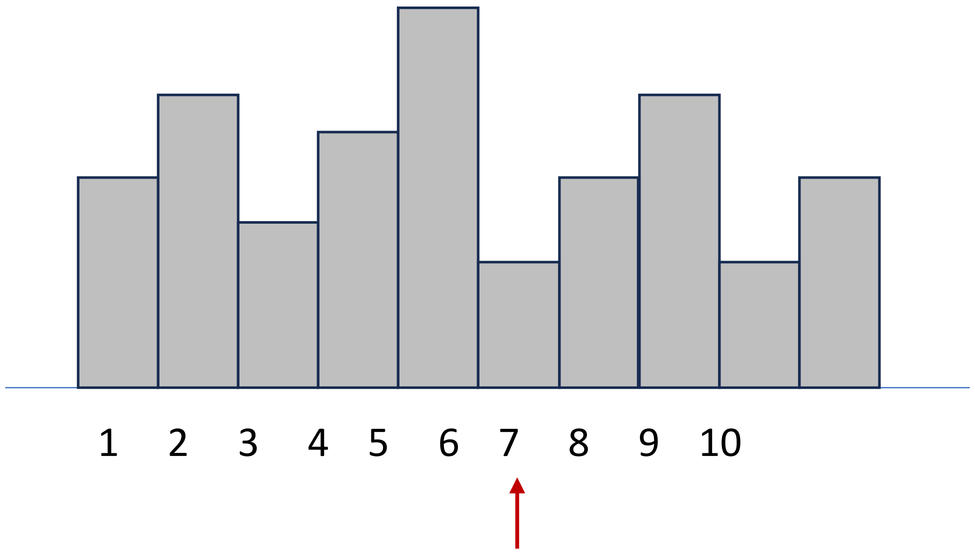
$R[2] = 3$

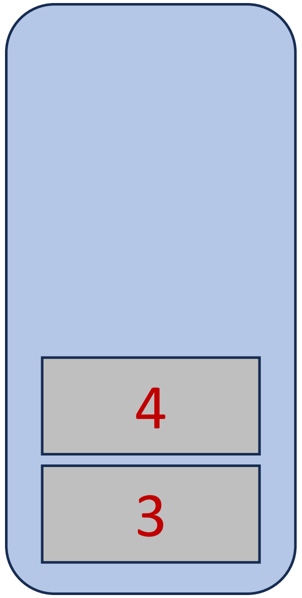
$R[1] = 3$



Method

- ➢ Solving histogram problem
- Compute $R[i]$:
    - For each column j:
        - while $h[j] < h[S.top]$ do
            - pop an index i out of S $(i = S.top)$
            - assign $R[i] = j$
        - Push j into S



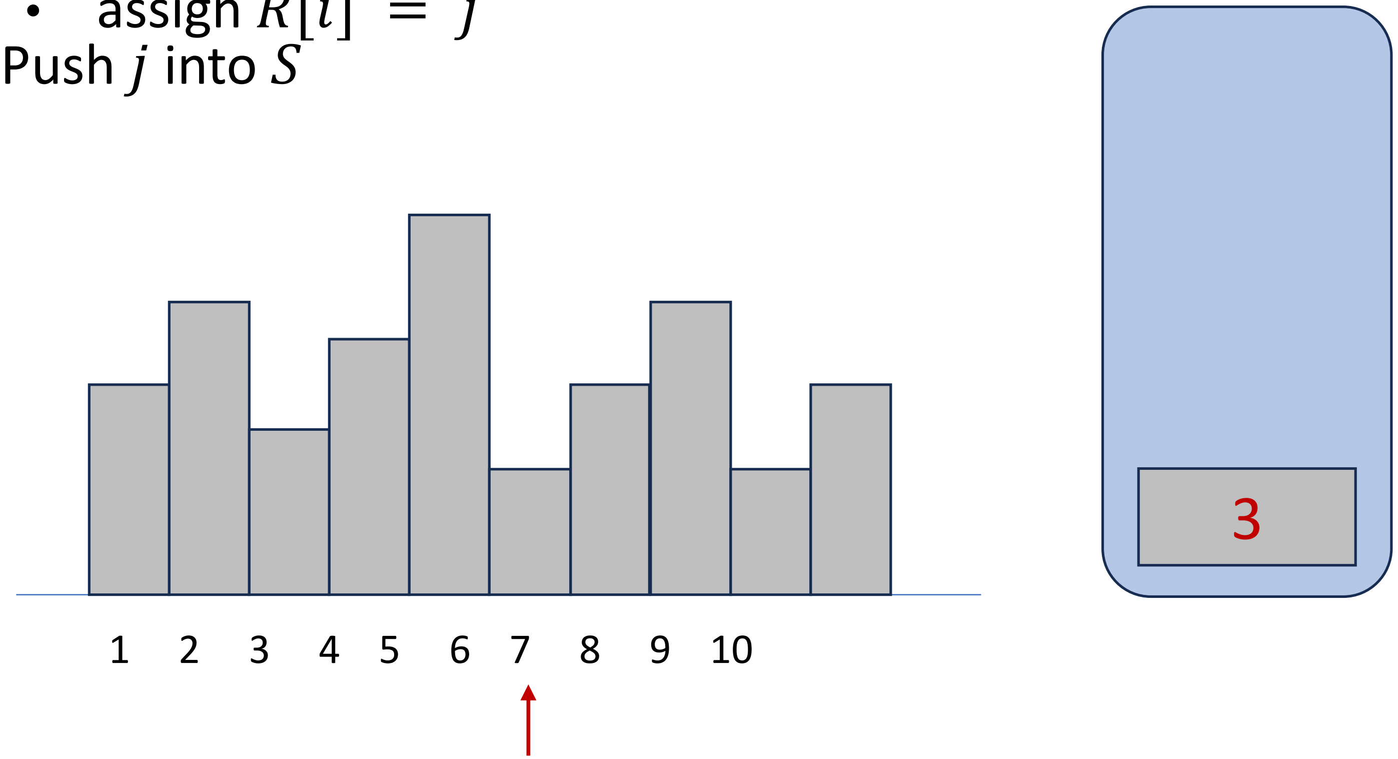


$R[2] = 3$
$R[1] = 3$
$R[5] = 6$

Method

- ➢ Solving histogram problem

```
Compute $R[i]$:
    For each column $j$:
        while $h[j] < h[S.top]$ do
            pop an index $i$ out of $S$ ($i = S.top$)
            assign $R[i] = j$
        Push $j$ into $S$
```



$R[2] = 3$

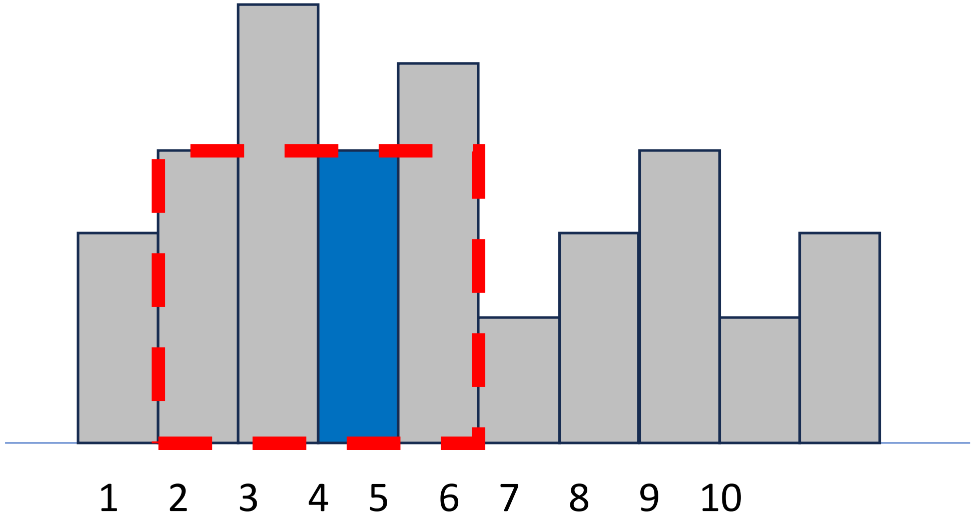
$R[1] = 3$

$R[5] = 6$

$R[4] = 6$

Method

- ➢ Solving histogram problem
- Compute $L[i]$:
  - Explore columns from right to left, maintain a stack S containing indices i of columns waiting for the computation of $L[i]$
  - For each column j:
    - while h[j] < h[S.top] do
      - pop an index i out of S $(i = S.top)$
      - assign $L[i] = j$
    - Push j into S



HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !