

85 : Maximal Rectangle Leetcode Hard



Rohith Vazhathody · [Follow](#)

4 min read · Apr 6, 2021



Listen



Share

85. Maximal Rectangle

Hard



4081



86



Add to List



Share

Given a `rows x cols` binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return *its area*.

Example 1:

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

Leetcode 85 : Maximal Rectangle

Problem Statement :

Given a `rows x cols` binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return *its area*.

Sample Test Cases :

Example 1:

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

Sample Visualisation for example 1

Input: `matrix = [["1","0","1","0","0"],["1","0","1","1","1"],`
`["1","1","1","1","1"],["1","0","0","1","0"]]`

Output: 6

Explanation: The maximal rectangle is shown in the above picture.

Example 2:

Input: `matrix = []`

Output: 0

Example 3:

Input: matrix = `[["0"]]`

Output: 0

Example 4:

Input: matrix = `[["1"]]`

Output: 1

Example 5:

Input: matrix = `[["0","0"]]`

Output: 0

Constraints :

- rows == matrix.length
- cols == matrix[i].length
- 0 <= row, cols <= 200
- matrix[i][j] is '0' or '1'.

Approach :

After reading the statement, we may assume this is similar to another problem that is maximal square (Leetcode : 221 Maximal Square). But the way we need to do this problem is somewhat different from Maximal Square. Yes, we need to use dynamic programming but we also need to use the exact concept as that of another problem Largest Rectangle in Histogram.

Here we only need to consider the 1's and need to find the largest rectangle possible.

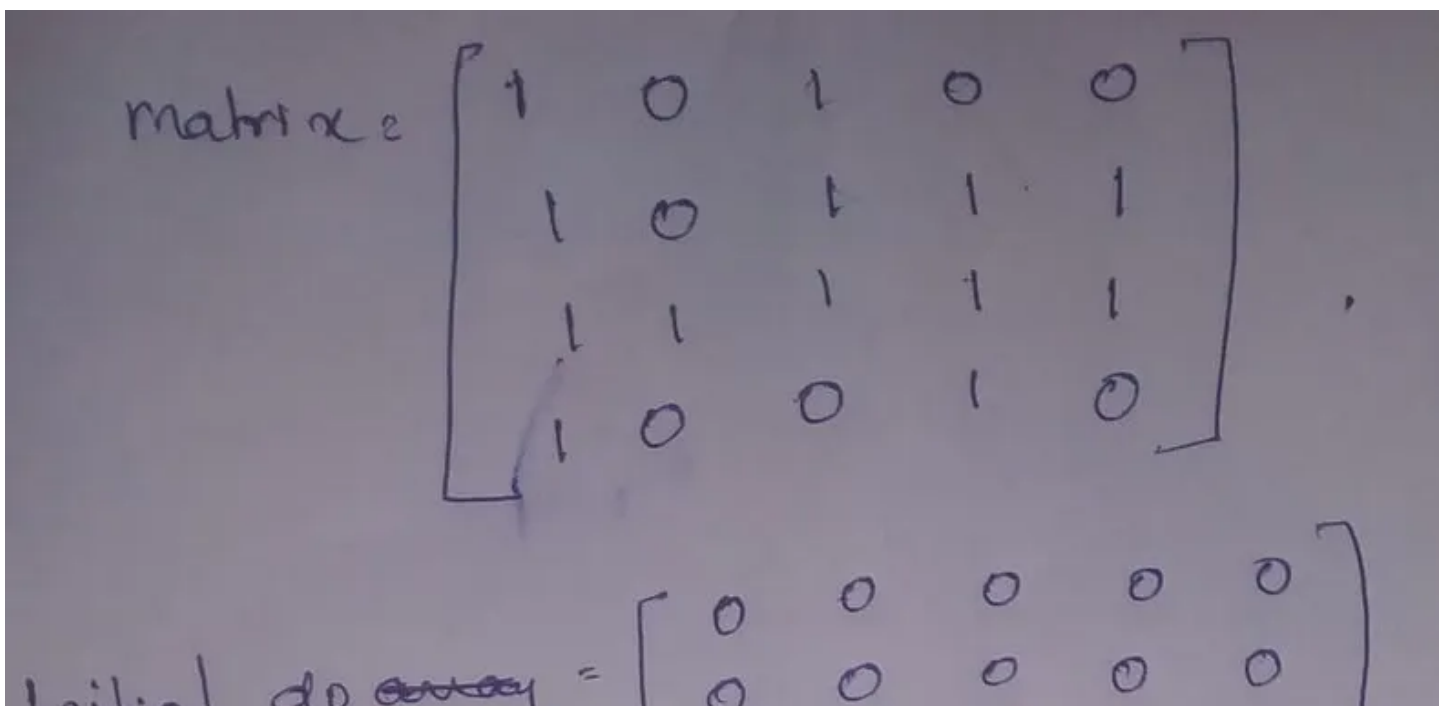
The first thing to do is create a dp matrix of size row and col which is exactly same as the input matrix.

Now we need to populate the dp matrix with the heights. How we are populating with the height?

Say we are now currently on a cell say (i, j) which is 1. We look on the top of the cell ie. (i-1, j) and check whether it is one too. If it is 1, then we update the value of cell(i, j) as $\text{valueOf}(i-1, j) + \text{inputMatrixCellValueOf}(i, j)$.

Same way, if we are at the cell (i, j) which is 0. Then no matter what is the value on the top, the value we are populating on the dp matrix will be 0 as the height cannot be updated if we have a value 0.

```
for (int i=0; i<row; i++) {
    for (int j=0; j<col; j++) {
        if (i == 0) {
            dp[i][j] = matrix[i][j] - '0';
        }
        else {
            if (matrix[i][j] == '1')
                dp[i][j] = dp[i - 1][j] + matrix[i][j] -
'0';
            }
        }
    }
```



Populating dp matrix by height i

$$dp = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1+1 & 0 & 1+1 & 1+0 & 1+0 \\ 1+1+1 & 1+0 & 1+1+1 & 1+1 & 1+1 \\ 1+1+1+1 & 0 & 0 & 1+1+1 & 0 \end{bmatrix}$$

$$r = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 2 & 0 & 2 & 1 & 1 \\ 3 & 1 & 3 & 2 & 2 \\ 4 & 0 & 0 & 3 & 0 \end{bmatrix}$$

Open in app ↗

Sign up

Sign in

●● Medium



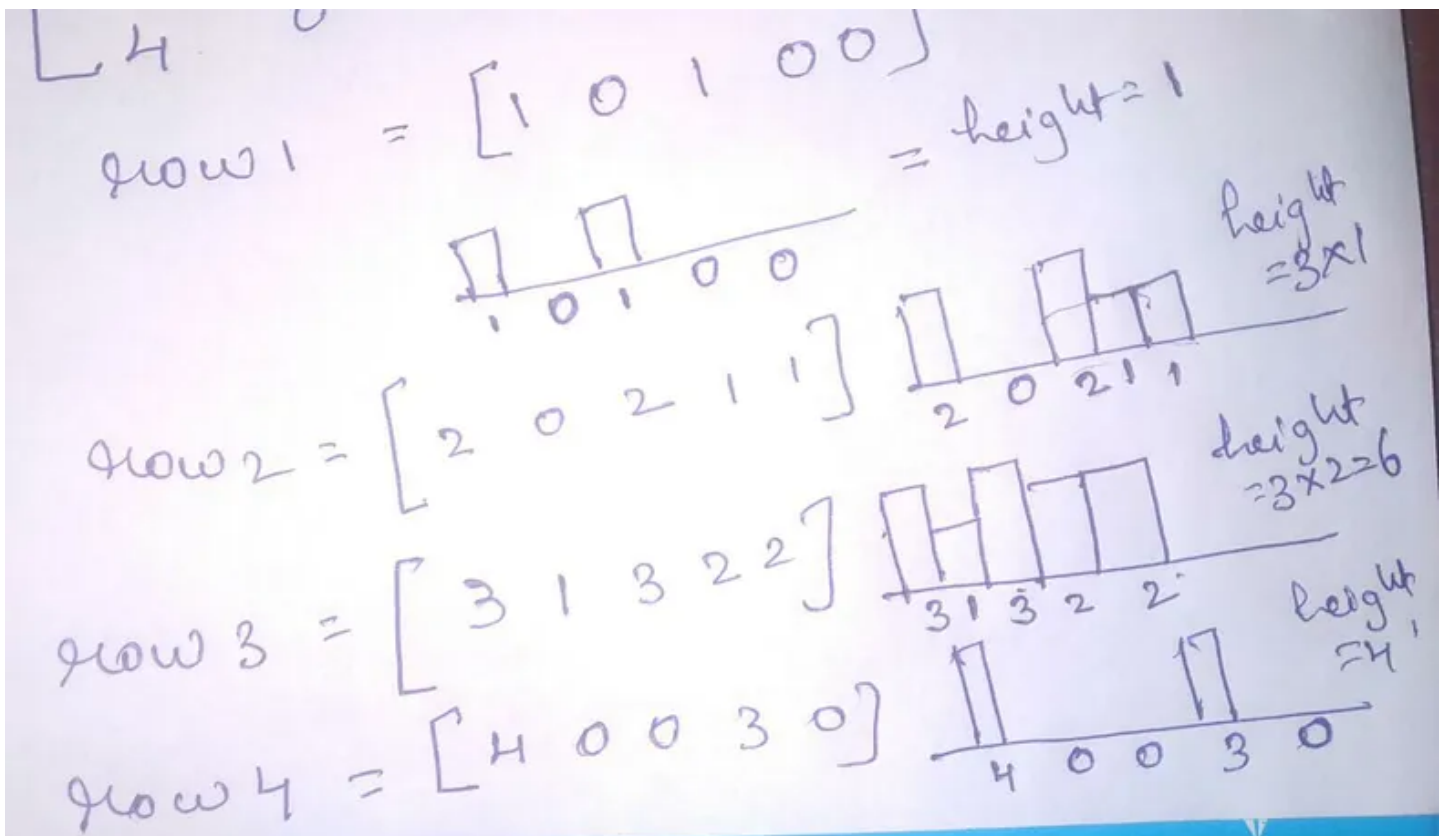
the largest rectangle from each individual row. After processing each row, then we have the maximal rectangle.

```
// same code for largest rectangle in a histogram
public int largestRectangleHistogram(int [] h) {
    if (h == null || h.length == 0)
        return 0;
    int length = h.length;
```

```

int [] left = new int [length];
int [] right = new int [length];
int max = 0;
left[0] = -1;
right[length - 1] = length;
for (int i=1; i<length; i++) {
    int currentIndex = i - 1;
    while (currentIndex >= 0 && h[currentIndex] >= h[i]) {
        currentIndex = left[currentIndex];
    }
    left[i] = currentIndex;
}
for (int i=length - 2; i>=0; i--) {
    int currentIndex = i + 1;
    while (currentIndex < length && h[currentIndex] >= h[i])
    {
        currentIndex = right[currentIndex];
    }
    right[i] = currentIndex;
}
for (int i=0; i<length; i++) {
    max = Math.max(max, h[i] * (right[i] - left[i] - 1));
}
return max;

```



Thus we find the maximum height from each row and the maximum of those is the answer we are looking.

Complete Code :

```
class Solution {
    public int maximalRectangle(char[][] matrix) {
        int row = matrix.length;
        if (row == 0)
            return 0;
        int col = matrix[0].length;
        int max = 0;
        int [][] dp = new int [row][col];
        // calculate the height
        for (int i=0; i<row; i++) {
            for (int j=0; j<col; j++) {
                if (i == 0) {
                    dp[i][j] = matrix[i][j] - '0';
                }
                else {
                    if (matrix[i][j] == '1')
                        dp[i][j] = dp[i - 1][j] + matrix[i][j] -
'0';
                }
            }
        }
        for (int [] eachRow : dp) {
            max = Math.max(max, largestRectangleHistogram(eachRow));
        }
        return max;
    }

    // same code for largest rectangle in a histogram
    public int largestRectangleHistogram(int [] h) {
        if (h == null || h.length == 0)
            return 0;
        int length = h.length;
        int [] left = new int [length];
        int [] right = new int [length];
        int max = 0;
```

```

left[0] = -1;
right[length - 1] = length;
for (int i=1; i<length; i++) {
    int currentIndex = i - 1;
    while (currentIndex >= 0 && h[currentIndex] >= h[i]) {
        currentIndex = left[currentIndex];
    }
    left[i] = currentIndex;
}
for (int i=length - 2; i>=0; i--) {
    int currentIndex = i + 1;
    while (currentIndex < length && h[currentIndex] >= h[i]) {
        currentIndex = right[currentIndex];
    }
    right[i] = currentIndex;
}
for (int i=0; i<length; i++) {
    max = Math.max(max, h[i] * (right[i] - left[i] - 1));
}
return max;
}
}

```

Time Complexity and Space Complexity:

Time : $O(\text{row} * \text{col})$

Space : $O(\text{row} * \text{col})$ new dp array

Github Repo :

Rohithv07/LeetCode

LeetCode problems that are solved. Contribute to Rohithv07/LeetCode development by creating an account on...

github.com

Rohithv07/LeetCode

LeetCode problems that are solved. Contribute to Rohithv07/LeetCode development by creating an account on...

github.com

Rohithv07/LeetCodeTopInterviewQuestions

Leetcode Top Interview questions discussed in Leetcode...

github.com

Data Structures

Interview Questions

Leetcode

Algorithms

Dynamic Programming



Follow

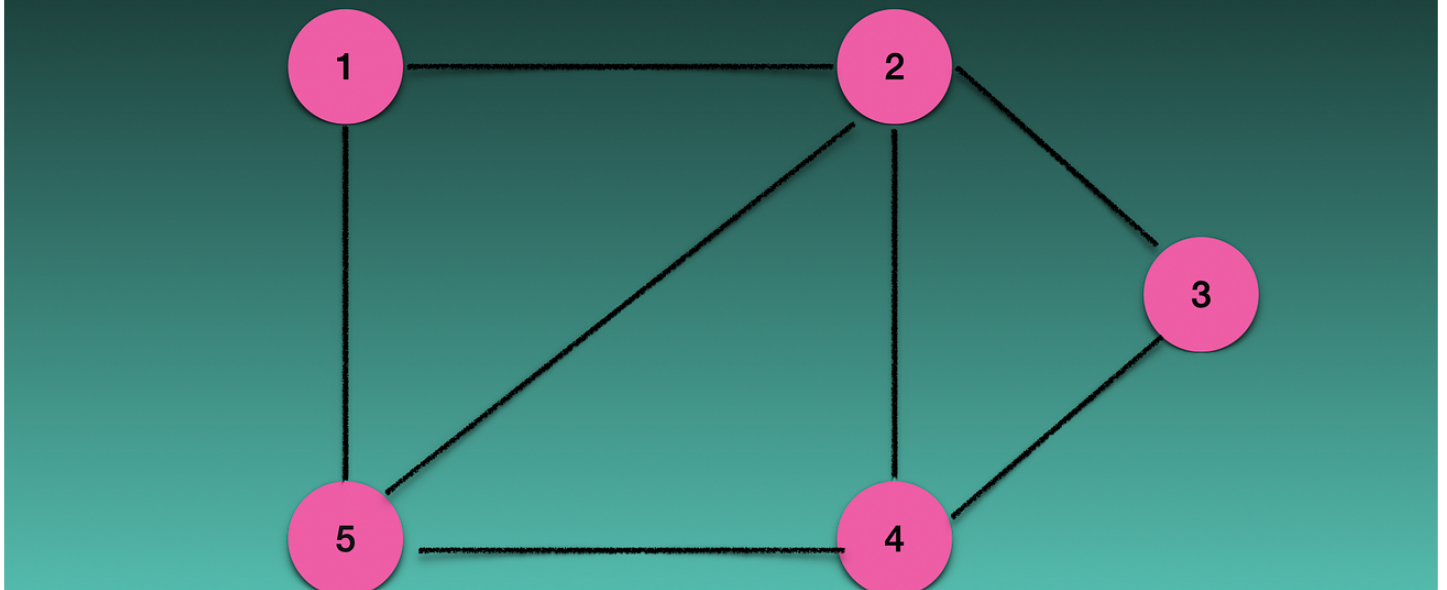
Written by Rohith Vazhathody

189 Followers

Software Engineer | Weekly articles | Interested in DSA, design | Writes about problem solving, algorithm explanation of my understanding and Java Codes.

More from Rohith Vazhathody

Bipartite Graph-DFS Graph Coloring



Rohith Vazhathody

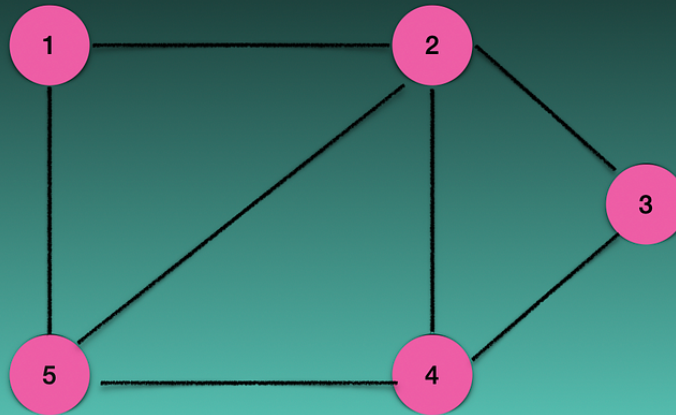
Graph Algorithm—Bipartite Graph(DFS)

What is a Bipartite Graph

3 min read · Apr 17, 2022



Bipartite Graph-BFS Graph Coloring



 Rohith Vazhathody

Graph Algorithm—Bipartite Graph(BFS)

What is a Bipartite Graph

4 min read · Apr 15, 2022

 3 



Example 1:

Input: `s = "(()"`

Output: 2

Explanation: The longest valid

parentheses substring is "()"

parentheses substring is "()".

 Rohith Vazhathody

32. Longest Valid Parentheses Leetcode Dynamic Programming

Problem Statement :

3 min read · Apr 5, 2021

 2 



Description

Solution

Discuss (999+)

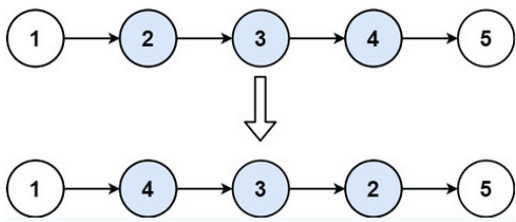
Submissions

92. Reverse Linked List II

Medium 7213 318 Add to List Share

Given the head of a singly linked list and two integers left and right where left ≤ right, reverse the nodes of the list from position left to position right, and return the reversed list.

Example 1:



Input: head = [1,2,3,4,5], left = 2, right = 4
Output: [1,4,3,2,5]

Example 2:

Input: head = [5], left = 1, right = 1
Output: [5]

Java

Autocomplete

```
1  /**
2   * Definition for singly-linked list.
3   * public class ListNode {
4   *     int val;
5   *     ListNode next;
6   *     ListNode() {}
7   *     ListNode(int val) { this.val = val; }
8   *     | ListNode(int val, ListNode next) { this.val = val; this.next = next; }
9   * }
10  */
11  class Solution {
12  public ListNode reverseBetween(ListNode head, int left, int right) {
13      if (head == null || head.next == null) {
14          return head;
15      }
16      ListNode previous = null;
17      ListNode current = head;
18      while (left > 1) {
19          previous = current;
20          current = current.next;
21          left--;
22          right--;
23      }
24      ListNode connection = previous;
25      ListNode tail = current;
26      while (right > 0) {
27          ListNode next = current.next;
28          current.next = previous;
29          previous = current;
30      }
31      if (connection != null) {
32          connection.next = tail;
33      }
34      return head;
35  }
36  }
```

Your previous code was restored from your local storage. [Reset to default](#)

 Rohith Vazhathody

Leetcode 92. Reverse Linked List II

Problem Statement

2 min read · Jul 21, 2022

 1 



See all from Rohith Vazhathody

Recommended from Medium



Jackie Nguyen


Leetcode 1143. Longest Common Subsequence

Bottom Up

2 min read · Mar 11, 2024





 B4night

Leetcode 2870: Use DP to memory minimum steps.

First of all, we can use map to record each number and their appearing times.

1 min read · Jan 3, 2024



Lists



Practical Guides to Machine Learning

10 stories · 1371 saves



General Coding Knowledge

20 stories · 1166 saves



Productivity

240 stories · 416 saves



Staff Picks

630 stories · 928 saves






 金士淳

LeetCode No1.Two Sum

Introduce

3 min read · Dec 13, 2023

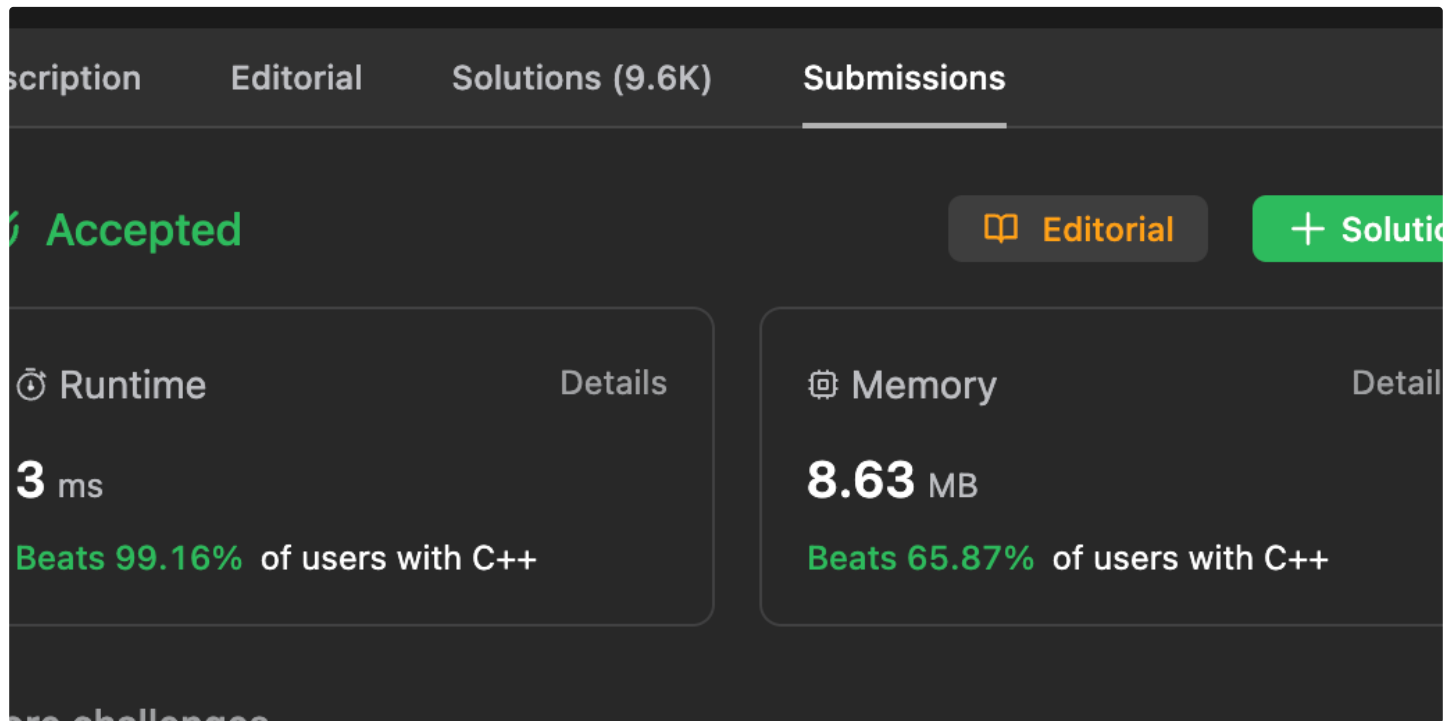


 Opeyemi Olorunleke

House Robber || Leetcode

Problem can be found here

4 min read · Dec 29, 2023



description Editorial Solutions (9.6K) Submissions

Accepted Editorial + Solutions

Runtime Details

3 ms

Beats 99.16% of users with C++

Memory Detail

8.63 MB

Beats 65.87% of users with C++

 Nicholas Anderson in Rakulee

5. Longest Palindromic Substring

<https://leetcode.com/problems/longest-palindromic-substring/>


5 min read · Nov 1, 2023



21





 Chandan Naik

Leetcode 2870

Minimum Number of Operations to Make Array Empty

3 min read · Jan 3, 2024



See more recommendations