48. Rotate Image

> **Input:** 1 2 3 4 5 6 7 8 9
> **Output:** 7 4 1 8 5 2 9 6 3

> **Input:** 1 2 3 4 **Output:** 3 1 4 2

Let size of row and column be 3.

> During first iteration –

- a[i][j] = Element at first index (leftmost corner top)= 1.
- a[j][n-1-i]= Rightmost corner top Element = 3.
- a[n-1-i][n-1-j] = Rightmost corner bottom element = 9.
- a[n-1-j][i] = Leftmost corner bottom element = 7. Move these elements in the clockwise direction.

  > During second iteration –

- a[i][j] = 2.
- a[j][n-1-i] = 6.
- a[n-1-i][n-1-j] = 8.
- a[n-1-j][i] = 4. Similarly, move these elements in the clockwise direction.

GFG (https://www.geeksforgeeks.org/rotate-a-matrix-by-90-degree-in-clockwise-direction-without-using-any-extra-space/)

---

50. Pow(x, n)

Steps:

1. First check if the given number is less the Integer.MIN_VALUE or greater than the Integer.MAX_VALUE and the abs(power) is greater than 1.
2. If the `step 1` satisfies then return 0.
3. Now check if the power is negative, if negative then convert it to positive and number to its reciprocal.

   ```
   `Because x power +ve power is same as 1/x power -ve power.`
   ```

4. Run a loop until the power is not zero. i. Check if the power is odd, if odd then multiply the x with the ans.

   ```
   ii. If the power id even then multiply x with itself and half the power n.
         `This will reduce the number of multiplications.`
   ```

5. Return the result double ans.

```
class Solution {
    public double myPow(double x, int n) {
        double ans = 1.0;
        if((n <= Integer.MIN_VALUE ||  n >= Integer.MAX_VALUE) && Math.abs(x) != 
1) return ans*0;
        if(n < 0){
            n = -n;
            x = 1 / x;
        }
        while(n > 0) {
            if((n & 1) != 0) {
                ans *= x;
                n--;
            } else {
                x *= x;
                n >>>= 1;
            }
        }

        return ans;
    }
}
```

73. Set Matrix Zeroes

Method 1: (Brute force) -using another matrix (let's say it matrix2)

we can copy all the elements of given matrix to matrix2 while traversing given matrix whenever we encounter 0, we will make the entire row and column of the matrix2 to 0 finally we can again copy all the elements of matrix2 to given matrix -Time: O((mn)*(m+n)), Space: O(mn) image

```
public void setZeroes(int[][] matrix){

        int m= matrix.length, n= matrix[0].length;
        int matrix2[][]= new int[m][n];
        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++)
                matrix2[i][j]=matrix[i][j];
        }

        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++){
                if(matrix[i][j]==0){
                    for(int k=0;k<n;k++)
                        matrix2[i][k]=0;

                    for(int k=0;k<m;k++)
                        matrix2[k][j]=0;
                }
            }
        }

        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++)
                matrix[i][j]=matrix2[i][j];
        }
    }

    ```
Method 2: (Better)

we can use two separate arrays one for rows (rowsArray) and one for columns (colsA
rray) and initialize them to 1
while traversing the given matrix whenever we encounter 0 at (i,j), we will set ro
wsArray[i]=0 and colsArray[j]=0
After completion of step 2, again iterate through the matrix and for any (i,j), if
rowsArray[i] or colsArray[j] is 0 then update matrix[i][j] to 0.
-Time: O(mn), Space: O(m+n)
image
```

public void setZeroes(int[][] matrix){
```

```
    int m=matrix.length, n=matrix[0].length;
    int rowsArray[]= new int[m];
    int colsArray[]= new int[n];

    Arrays.fill(rowsArray,1);
    Arrays.fill(colsArray,1);

    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            if(matrix[i][j]==0){
                rowsArray[i]=0;
                colsArray[j]=0;
            }
        }
    }

    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            if(rowsArray[i]==0 || colsArray[j]==0)
                matrix[i][j]=0;
        }
    }
}
```

```
Method 3: (Optimal)
-we can use the 0th row and 0th column of the given matrix itself instead of using
two separate arrays

first we will traverse the 0th row and 0th column of the given matrix and if we en
counter any 0 then we will set the isRow0/isCol0 variable to true which indicates
that the 0th row/0th column of the given matrix will become 0
next we will traverse the remaining matrix except 0th row and 0th column and if we
encounter any 0, we will make the corresponding row no. and column no. equal to 0
in the 0th column and 0th row respectively
Now we will update the values of the matrix except first row and first column to 0
if matrix[i][0]=0 or matrix[0][j]=0 for any (i,j).
finally we will traverse the 0th row and 0th column and if we find any 0, we will
make the whole row and whole column equal to 0
-Time: O(mn), Space: O(1)
image
```

public void setZeroes(int[][] matrix){

```
    int m=matrix.length, n=matrix[0].length;
    boolean isRow0=false, isCol0=false;

    for(int j=0;j<n;j++){
        if(matrix[0][j]==0)
            isRow0=true;
    }

    for(int i=0;i<m;i++){
        if(matrix[i][0]==0)
            isCol0=true;
    }

    for(int i=1;i<m;i++){
        for(int j=1;j<n;j++){
            if(matrix[i][j]==0){
                matrix[i][0]=0;
                matrix[0][j]=0;
            }
        }
    }

    for(int i=1;i<m;i++){
        for(int j=1;j<n;j++){
            if(matrix[0][j]==0 || matrix[i][0]==0)
                matrix[i][j]=0;
        }
    }

    if(isRow0){
        for(int j=0;j<n;j++)
            matrix[0][j]=0;
    }

    if(isCol0){
        for(int i=0;i<m;i++)
            matrix[i][0]=0;
    }
}
```

``` Don't forget to upvote if you find it helpful! 👍

## 88. Merge Sorted Array
```

83. Merge Sorted Array

```
int i = m - 1;     // nums1's index (actual nums)
int j = n - 1;     // nums2's index
int k = m + n - 1; // nums1's index (next filled position)

while (j >= 0)
    if (i >= 0 && nums1[i] > nums2[j])
        nums1[k--] = nums1[i--];
    else
        nums1[k--] = nums2[j--];
```

- we need to fill from backwords in the nums1
- we need to do this until elements of nums2 exist
- This replaces the zeros at the end nedds tobe filled

---

142. Linked List Cycle II

---

169. Majority Element

169. Majority Element

- loop through the numbers.
- initially take one number and start counting it the number is same as result
- if not decrement the mode
- if the mode is zero then other number is repeating more times than this number
- so change result to this number
- and continue the above process

  ```
  public int majorityElement(int[] nums) {
              int res = 0, mode = 0;
              for (int num : nums) {
                      if (mode == 0) {res = num; ++mode;}
                      else if (num == res) ++mode;
                      else --mode;
              }
              return res;
      }
  ```

---

237. Delete Node in a Linked List

- Make the current node value to the next node value.
- And replace the next node with its next node.

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public void deleteNode(ListNode node) {
        node.val = node.next.val;
        node.next = node.next.next;
    }
}
```

450. Delete Node in a BST

✓ JAVA | Recursive | Most Intutive | Proper Explanation Using Image Chaitanya1706 2606 6460 Nov 22, 2021 Java Recursion Intution: Use the BST property to search the node and then delete if found the required node.

So if the traget node has value less than root then we will surely get it in the left subtree...so just call ur recursive function for the left subtree. If the traget node has value greater than root then we will surely get it in the right subtree...so just call ur recursive function for the right subtree. And now comes the case when u have to do your work that is root itself is the required node to be deleted. Here again comes three cases: If left of root is null and u also have to delete the root node...then just simply return the right subtree.

If right of root is null and u also have to delete the root node...then just simply return the left subtree.

Both are not null then you have to not just delete the node but also maintain the BST structure. So now you have to think if you delete the root node then which node can optimally replace it so that all the nodes on left are still small and on right are larger. So that node will be the node just greater than the largest node in the left subtree which is the smallest node in the right subtree image

So point your pointer on the right subtree and then move it to the left most node of this subtree that will be your required node and so now replace the value of your root with this node value which will ensure that the key which u wanted to delete is deleted and the value there is the right value. Now you have to delete that node whose value is already present in the root...so now that work will be done by the recursion so now just pass that right subtree in which the value is present with that nodes value which will be now the target

```java
class Solution {
    public TreeNode deleteNode(TreeNode root, int key) {
        if(root==null) return null;

        if(key<root.val){
            root.left = deleteNode(root.left,key);
            return root;
        }

        else if(key>root.val){
            root.right = deleteNode(root.right,key);
            return root;
        }

        else{
            if(root.left==null){
                return root.right;
            }
            else if(root.right==null){
                return root.left;
            }
            else{
                TreeNode min = root.right;
                while(min.left!=null){
                    min = min.left;
                }

                root.val = min.val;
                root.right = deleteNode(root.right,min.val);
                return root;
            }
        }
    }
}
```