LAB-02

TASK-01:

1.QUICK SORT(pivot at end)-

```
#include <iostream>
using namespace std;
int partition(int arr[],int low,int high){
    int pivot = arr[high];
    int i = low-1;

    for(int j= low; j<high ; j++){
        if(pivot>arr[j]){
            i++;
            swap(arr[i],arr[j]);
        }
    }
    swap(arr[i+1] , arr[high]);
    return i+1;
}
void quicksort(int arr[] , int low , int high){
    if(low<high){
        int pivot = partition(arr,low,high);

        quicksort(arr,low,pivot-1);
        quicksort(arr,pivot+1,high);
    }
}
void print(int arr[],int n){
    for(int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
int main(){
    int arr[5] = {3,1,5,6,2};
    int n = 5;
    print(arr,n);
    quicksort(arr,0,n-1);
    cout<<"Pivot at end: ";
    print(arr,n);
}
```

Output:

```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2> cd "c:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2\
 }
3 1 5 6 2
Pivot at end: 1 2 3 5 6
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2>
```

2.Quick Sort(pivot at first):-

```
#include <iostream>
using namespace std;
int partition(int arr[],int low,int high){
    int pivot = arr[low];
    int j = high;
    for (int i = high; i > low; i--) {
        if (arr[i] > pivot){
```

```
            swap(arr[i],arr[j]);
            j--;
        }
    }
    swap(arr[j],arr[low]);
    return j;
}
void quicksort(int arr[] , int low , int high){
    if(low<high){
        int pivot = partition(arr,low,high);

        quicksort(arr,low,pivot-1);
        quicksort(arr,pivot+1,high);
    }
}
void print(int arr[],int n){
    for(int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
int main(){
    int arr[5] = {3,1,5,6,2};
    int n = 5;
    print(arr,n);
    quicksort(arr,0,n-1);
    cout<<"Pivot at start: ";
    print(arr,n);
}
```

Output:

```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2> cd "c:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2\" ; i
 }
13 1 15 6 2
Pivot at start: 1 2 6 13 15
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2>
```

3.QuickSort(pivot at middle)

```
#include <iostream>
using namespace std;
int partition(int arr[],int low,int high){
    int mid = (low + high)/2;
    int pivot = arr[mid];
    swap(arr[mid] , arr[high]);
    int i = low;

    for(int j= low; j<high ; j++){
        if(pivot>arr[j]){

            swap(arr[i],arr[j]);
            i++;
        }
    }
    swap(arr[i] , arr[high]);
    return i;
}
void quicksort(int arr[] , int low , int high){
    if(low<high){
        int pivot = partition(arr,low,high);

        quicksort(arr,low,pivot-1);
```

```
        quicksort(arr,pivot+1,high);
    }
}
void print(int arr[],int n){
    for(int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
int main(){
    int arr[5] = {3,1,5,6,2};
    int n = 5;
    print(arr,n);
    quicksort(arr,0,n-1);
    cout<<"Pivot at end: ";
    print(arr,n);
}
```

Output:

```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2> cd "c:\Users\R
 }
3 9 10 1 4
Pivot at middle: 1 3 4 9 10
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2>
```

Time complexities:

Pivot	Start	Middle	End
Random	n*logn	n*logn	n*logn
Best	n*logn	n*logn	n*logn
Worst	n^2	n^2	n^2

4.Decimal to Binary:

```
#include <bits/stdc++.h>
using namespace std;
void find_bin(int n){
    if(n==0){
        return ;
    }
    find_bin(n/2);
    cout<<n % 2;
}
int main(){
    int n;
    cout<<"Enter any number: ";
    cin>>n;
    int p=n;
    string binary = "";
    if (n == 0) {
        binary = "0";
    } else {
        while (n > 0) {
            binary = char('0' + (n % 2)) + binary;
            n /= 2;
        }
    }
    cout<<"Binary from loop: "<<binary<<endl;
    cout<<"Binary from Recurssion: ";
```

```
    find_bin(p);

}
```

Output:

```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2> cd "c:\Users\Ritik gupta\Desktop\Lab\DAA\
 }
Enter any number: 22
Binary from loop: 10110
Binary from Recurssion: 10110
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 2>
```

Task-02

1.Number of Islands:

Description | Accepted | Editorial | Solutions | Submissions

200. Number of Islands

Solved

Medium | Topics | Companies

Given an `m x n` 2D binary grid `grid` which represents a map of `'1'`s (land) and `'0'`s (water), return *the number of islands.*

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

```
Input: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
Output: 1
```

Example 2:

23.4K | 239 | 356 Online

Code

C++ | Auto

```cpp
public:
    void island(vector<vector<char>>& grid , int i , int j , int row , int col){
        if(i<0 || j<0 || i>=row || j>=col|| grid[i][j] == '0')return;
        grid[i][j] = '0';
        island(grid,i+1,j,row,col);
        island(grid,i,j+1,row,col);
        island(grid,i-1,j,row,col);
        island(grid,i,j-1,row,col);
    }

    int numIslands(vector<vector<char>>& grid) {
        int row = grid.size();
        int col = grid[0].size();
        int count =0;
        for(int i =0;i<row;i++){
            for(int j =0; j<col;j++){
                if(grid[i][j] == '1'){
                    island(grid,i,j,row,col);
                    count++;
                }
            }
        }
        return count;
```

Saved | Ln 4

Testcase | Test Result

Accepted 49 / 49 testcases passed

RITIK GUPTA submitted at Jan 29, 2025 12:24

Editorial | Solution

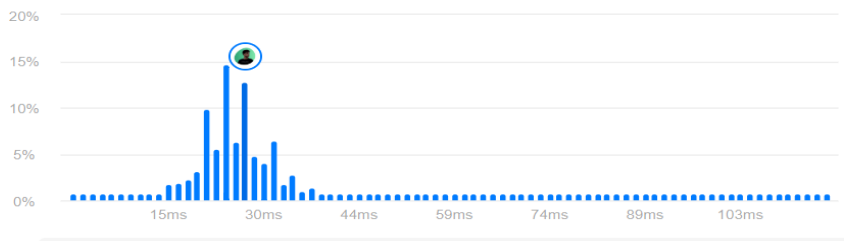
Runtime

27 ms | Beats 58.47%

Analyze Complexity

Memory

16.19 MB | Beats 89.28%



2.Merge 2 sorted array

Description | Editorial | Solutions | Submissions

88. Merge Sorted Array

Solved

Easy | Topics | Companies | Hint

You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to `0` and should be ignored. `nums2` has a length of `n`.

Code

C++ | Auto

```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int first = m-1;
        int second = n-1;
        int idx = m+n-1;
        while(second>=0){
            if(first>=0 && nums1[first] > nums2[second]){
                nums1[idx--] = nums1[first--];
            }
            else{
                nums1[idx--] = nums2[second--];
            }
        }
    }
```

Accepted 59 / 59 testcases passed

RITIK GUPTA submitted at Jan 31, 2025 18:59

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

12.20 MB | Beats 70.85%

100%

50%

3.Length of last word

Description | Editorial | Solutions | Submissions

58. Length of Last Word

Solved

Easy | Topics | Companies

Given a string `s` consisting of words and spaces, return *the length of the* ***last*** *word in the string.*

A **word** is a maximal substring consisting of non-space characters only.

Example 1:

```
Input: s = "Hello World"
```

Code

C++ | Auto

```cpp
class Solution {
public:
    int lengthOfLastWord(string s) {

        int count=0;
        int i = s.length()-1;
        while (i >= 0 && s[i] == ' ') i--;
        while(i >= 0 && s[i] != ' '){i--;count++;}

        return count;
    }
};
```

All Submissions

Accepted 59 / 59 testcases passed

RITIK GUPTA submitted at Jan 28, 2025 09:41

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

8.95 MB | Beats 32.28%

100%

50%