

## Bubble Sort

```
class Solution {
public:
    // Function to sort the array using bubble sort algorithm.
    void bubbleSort(vector<int>& arr) {
        int n = arr.size();
        for (int i=0; i<n-1; i++){
            for (int j=0; j<(n-i-1); j++){
                if (arr[j]>arr[j+1]){
                    int temp = arr[j]; arr[j]=arr[j+1]; arr[j+1]=temp;
                }
            }
        }
    };
};
```

### Output:

The screenshot displays a coding platform interface. On the left, the 'Output Window' shows 'Compilation Results' for 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. Below this, statistics are shown: 'Test Cases Passed: 1115 / 1115', 'Attempts: 1 / 2', 'Accuracy: 50%', 'Points Scored: 2 / 2', and 'Time Taken: 0.05'. At the bottom, it shows 'Your Total Score: 38' with a green upward arrow. On the right, the code editor shows the same C++ code for Bubble Sort as provided in the previous block, with line numbers 1 through 25 visible.

Time Complexity:  $O(N^2)$

Space Complexity:  $O(1)$

## Quick Sort

### Output

Time Complexity:  $O(N * \log(N))$  (Worst case  $O(N^2)$  when in non increasing order)

Space Complexity:  $O(1)$

## First non repeating character

```
class Solution {
public:
```

```
// Function to find the first non-repeating character in a string.
char nonRepeatingChar(string &s) {
    vector<int> ctr (26,0);
    for (char i: s){
        ctr[i-97]++;
    }
    for (char i: s) {
        if (ctr[i-97]==1) return i;
    }
    return '$';
}
};
```

## Output

The screenshot shows a coding IDE with two main panels. The left panel, titled 'Output Window', displays 'Compilation Results' for a problem solved successfully. It shows 'Test Cases Passed: 1130 / 1130', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 2 / 2', and 'Time Taken: 0.04'. The right panel shows the source code for a C++ solution, which is identical to the code block provided above.

Time Complexity:  $O(N)$

Space Complexity:  $O(1)$

## Edit Distance

```
class Solution {
public:
    // Function to compute the edit distance between two strings
    int editDistance(string s1, string s2) {
        int m=s1.size(), n=s2.size();
        vector<vector<int>> dp (m+1,vector<int> (n+1,0));

        for (int i=1; i<=n; i++) dp[0][i] = i;
        for (int i=1; i<=m; i++) dp[i][0] = i;

        for (int i=1; i<=m; i++){
            for (int j=1; j<=n; j++){
                if (s1[i-1]!=s2[j-1]) dp[i][j] = min(min(dp[i][j-1], dp[i-1][j]), dp[i-1][j-1])+1;
                else dp[i][j]=dp[i-1][j-1];
            }
        }
    }
};
```

```

        /*
        for (vector<int> i: dp) {
            for (int j: i) cout<<j<<' ';
            cout<<endl;
        }*/
        return dp[m][n];
    }
};

```

## Output

The screenshot shows a coding platform interface. On the left, the 'Output Window' displays 'Compilation Results' for 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. Below this, it shows 'Test Cases Passed: 1115 / 1115', 'Attempts: Correct / Total: 1 / 4', 'Accuracy: 25%', 'Points Scored: 8 / 8', and 'Time Taken: 0.02'. At the bottom, it says 'Your Total Score: 32'. On the right, the code editor shows a C++ solution for the edit distance problem, using a 2D DP array.

Time Complexity:  $O(m * n)$

Space Complexity:  $O(m * n)$

## K largest elements

```

class Solution {
public:
    vector<int> kLargest(vector<int>& arr, int k) {
        priority_queue<int> pq;
        for (int i: arr) pq.push(i);
        vector<int> res;
        while (!pq.empty() && k--){
            res.push_back(pq.top());
            pq.pop();
        }
        return res;
    }
};

```

## Output:

```
1 // User function template for C++
2
3 // Problem Statement
4
5 // Solution
6
7 // User function template for C++
8 class Solution {
9 public:
10     vector<int> kLargest(vector<int>& arr, int k) {
11         priority_queue<int> pq;
12         for (int i: arr) pq.push(i);
13         vector<int> res;
14         while (!pq.empty() && k--){
15             res.push_back(pq.top());
16             pq.pop();
17         }
18         return res;
19     }
20 };
21 // Driver Code Ends
```

Time Complexity:  $O(k * \log(n))$

Space Complexity:  $O(n)$

## Form the largest number

```
class Solution {
public:
    string printLargest(vector<int> &arr) {
        int n = arr.size();
        vector<string> vs (n, "");
        for (int i=0; i<n; i++) vs[i]=to_string(arr[i]);
        sort(vs.begin(), vs.end(), [](string a, string b){
            if ((a+b)>(b+a)) return true;
            else return false;
        });
        string res = "";
        for (string i: vs) res+=i;
        return res;
    }
};
```

## Output:

```
1 // User function template for C++
2
3 // Problem Statement
4
5 // Solution
6
7 // User function template for C++
8 class Solution {
9 public:
10     string printLargest(vector<int> &arr) {
11         int n = arr.size();
12         vector<string> vs (n, "");
13         for (int i=0; i<n; i++) vs[i]=to_string(arr[i]);
14         sort(vs.begin(), vs.end(), [](string a, string b){
15             if ((a+b)>(b+a)) return true;
16             else return false;
17         });
18         string res = "";
19         for (string i: vs) res+=i;
20         return res;
21     }
22 };
23
```

Time Complexity:  $O(n * \log(n))$

Space Complexity:  $O(n)$