



PIMPRI CHINCHWAD EDUCATION TRUST'S.
PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
(An Autonomous Institute)

Class : SY BTech

Acad. Yr. 2025-26

Semester : I

Name of the student: Om Jitendra Khalane

PRN : 124B1B040

Department: Computer Engineering

Division : A

Course Name : Data Structures and Laboratory

Course Code: BCE23PC02

Completion Date : 23/07/2025

Assignment No. 02

Problem Statement:

A warehouse management system wants to sort inventory items by stock quantity to prioritize restocking. Write a program for above scenario.

Hint:

Given an unsorted list of inventory quantities, implement Quick Sort to sort items by stock quantity in ascending order. Discuss how the presence of many duplicate quantities affects Quick Sort's efficiency.

Source Code :

https://github.com/omkhalane/DSAL-SY-PCCOE/blob/main/lab_assignments/assignment02.cpp

```
#include <bits/stdc++.h>
using namespace std;

// partition and sort
int partition(vector<int> &arr, int low, int high)
{
    int pivot = arr[high];
    int i = low - 1;

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

```
        {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    return i + 1;
}

// quick sort
void quickSort(vector<int> &arr, int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main()
{
    vector<int> q;
    int n;

    cout << "Enter number of inventory items: ";
    cin >> n;

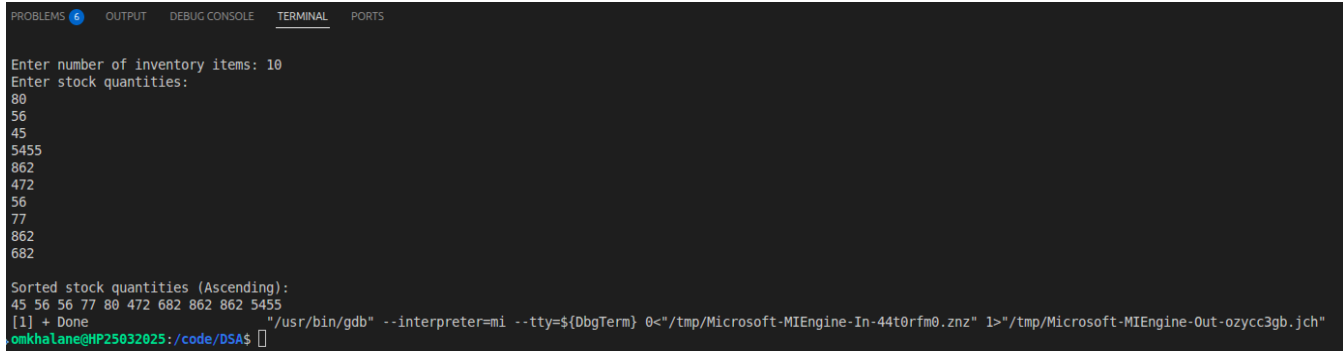
    cout << "Enter stock quantities:\n";
    for (int i = 0; i < n; i++)
    {
        int temp;
        cin >> temp;
        q.push_back(temp);
    }

    quickSort(q, 0, n - 1);

    cout << "\nSorted stock quantities (Ascending):\n";
    for (int i = 0; i < q.size(); i++)
    {
        cout << q[i] << " ";
    }
    cout << endl;
```

```
        return 0;
    }
```

Screen Shot of Output :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Enter number of inventory items: 10
Enter stock quantities:
80
56
45
5455
862
472
56
77
862
682

Sorted stock quantities (Ascending):
45 56 56 77 80 472 682 862 862 5455
[1] + Done
omkhalane@HP25032025:/code/DSA$
```

Conclusion:

Conclusion for the Quick Sort Program

- **Time Complexity (TC):**
 - **Best Case:** $O(n \log n)$ → Occurs when the pivot divides the array into two nearly equal halves each time.
 - **Average Case:** $O(n \log n)$ → Expected when elements are randomly distributed.
 - **Worst Case:** $O(n^2)$ → Happens when the pivot is always the smallest or largest element (e.g., sorted or reverse sorted data without randomization). This is more likely if there are **many duplicate values** and no special handling.
- **Space Complexity (SC):**
 - **Auxiliary Space:** $O(\log n)$ for the recursion stack in the best/average case, $O(n)$ in the worst case (highly unbalanced recursion).
 - **In-place Sorting:** No extra arrays are used for sorting, so apart from recursion stack, space usage is minimal.

Impact of Duplicates:

- When there are **many duplicate quantities**, Quick Sort's efficiency may drop because the pivot partitioning becomes unbalanced.

- This increases recursion depth, making the algorithm lean toward its worst-case $O(n^2)$ performance unless pivot selection is improved (e.g., using **randomized pivot** or **three-way partitioning**).