# DESIGN AND ANALYSIS OF ALGORITHMS

## LAB WORKBOOK WEEK-6

**NAME : M. Charitha Varshini**

**ROLL.NO: CH.SC.U4CSE24128**

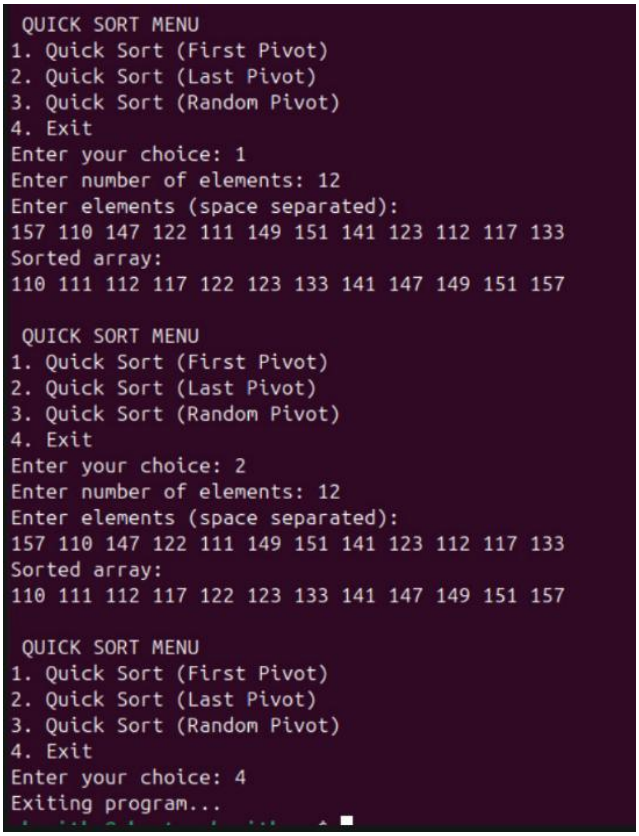
**DEPARTMENT: CSE-B**

## Quick sort taking First,Last,Random element as pivot elements

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void swap(int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}

/* Partition using LAST element */
int partitionLast(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;
}

/* FIRST element as pivot */
int partitionFirst(int arr[], int low, int high) {
    swap(&arr[low], &arr[high]);
```

```c
    return partitionLast(arr, low, high);
}

/* RANDOM pivot */
int partitionRandom(int arr[], int low, int high) {
    int r = low + rand() % (high - low + 1);
    swap(&arr[r], &arr[high]);
    return partitionLast(arr, low, high);
}

void quickSort(int arr[], int low, int high, int choice) {
    if (low < high) {
        int p;

        switch (choice) {
            case 1:
                p = partitionFirst(arr, low, high);
                break;
            case 2:
                p = partitionLast(arr, low, high);
                break;
            case 3:
                p = partitionRandom(arr, low, high);
                break;
            default:
                return;
        }
```

```
        }

        quickSort(arr, low, p - 1, choice);
        quickSort(arr, p + 1, high, choice);
    }
}

int main() {
    srand(time(NULL));

    int choice;

    do {
        printf("\n QUICK SORT MENU \n");
        printf("1. Quick Sort (First Pivot)\n");
        printf("2. Quick Sort (Last Pivot)\n");
        printf("3. Quick Sort (Random Pivot)\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice >= 1 && choice <= 3) {
            int n;
            printf("Enter number of elements: ");
            scanf("%d", &n);

            int arr[n];
            printf("Enter elements (space separated):\n");
            for (int i = 0; i < n; i++)
                scanf("%d", &arr[i]);

            quickSort(arr, 0, n - 1, choice);

            printf("Sorted array:\n");
            for (int i = 0; i < n; i++)
                printf("%d ", arr[i]);
            printf("\n");
        }
        else if (choice == 4) {
            printf("Exiting program...\n");
        }
        else {
            printf("Invalid choice! Try again.\n");
        }

    } while (choice != 4);

    return 0;
}
```




**TIME COMPLEXITY:- O(n log n)**

JUSTIFICATION:-

Quick Sort divides the array into two parts at each partition step. On average, the pivot divides the array into nearly equal halves. Each partition operation takes O(n) time to compare elements. The recursion depth is log n. Therefore, total time complexity = n × log n = O(n log n).

**SPACE COMPLEXITY :- O(log n)**

JUSTIFICATION :-
The given Quick Sort program uses recursion, hence extra space is required only for the recursion stack.
Each recursive call stores the following data:

int low → 4 bytes

int high → 4 bytes

int pivot → 4 bytes

return address and control information → 8 bytes

Total memory required per recursive call = 20 bytes

In the average case, the maximum recursion depth is log n.
Therefore, the total stack memory required is:

$$\mathbf{20 \times \log\ n} \text{ bytes}$$

Hence, the space complexity of the given code is O(log n).

Random pivot is better because it avoids unbalanced partitions, reduces the chance of worst-case time complexity, and gives consistent O(n log n)

performance for most inputs.

# Quick Sort

157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133

Pivot element = first element

i = (left + 1) element index

j = right index

Move i right side till A[i] > pivot

move j left side till A[j] < pivot

If i < j → Swap A[i], A[j]

If i ≥ j → Swap pivot with A[j]

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Array | 157 | 110 | 147 | 122 | 111 | 149 | 151 | 141 | 123 | 112 | 117 | 133 |

**Step 1:**

Pivot : 157

i=1 j=11

A[i] < pivot

A[j] > pivot

No element is greater than 157

j stays at 11

157 > 110 [move i right side]

157 > 147

157 > 122

⋮

157 > 133

i moves till end

i=j

so swap 157 ↔ 133

| 133 | 110 | 147 | 122 | 111 | 149 | 151 | 141 | 123 | 112 | 117 | 157 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

**Step 2:** Pivot (0-10)

Pivot = 133

i=1 j=10

133 > 110 ( i moves right)

133 < 147 (i stays at 147)

j=10

133 > 117 (j stays)

Swap (147, 117)

| 133 | 110 | 117 | 122 | 111 | 149 | 151 | 141 | 123 | 112 | 147 |
|---|---|---|---|---|---|---|---|---|---|---|

133 > 122 (moves i right)

133 > 111 (moves i right)

133 < 149 (stops at i=5)

112 < 133 (stops at j=9)

i < j

Swap(149, 112)

| 133 | 110 | 117 | 122 | 111 | 112 | 151 | 141 | 123 | 149 | 147 |
|---|---|---|---|---|---|---|---|---|---|---|

112 < 133 (moves i right)

151 > 133 (stops at j=6)

149 > 133 (moves j left)

123 < 133 (stops at j=8)

i < j

Swap(151, 123)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 133 | 110 | 117 | 122 | 111 | 112 | 123 | 141 | 151 | 149 | 147 |

123 < 133 (move i right)

141 > 133 (stops at i=7)

151 > 133 (moves j left)

141 > 133 (moves j left)

123 < 133 (stops at j=6)

i=7 j=6

i > j so swap (pivot, A[j])

(133, 123)

133 is fixed

| 123 | 110 | 117 | 122 | 111 | 112 | 133 | 141 | 151 | 149 | 147 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Step3:

Pivot = 123

Partition (0-5)

i=1 j=5

110 < 123 (i moves right)

117 < 123 ( " )

122 < 123 ( " )

111 < 123 ( " )

112 < 123 (at i=5)

j at 5

i=5 swap (A[j], pivot)

(112, 123)

| 112 | 110 | 117 | 122 | 111 | 123 |
|---|---|---|---|---|---|

123 fixed

Step 4

pivot = 112

partition (0-4)

| 112 | 110 | 117 | 122 | 111 |
|---|---|---|---|---|
| | ↑ i | | | ↑ j |

i=1 j=4

110 < 112 (i moves right)

117 > 112 (stops at i=2)

swap (117, 111)

j stops at 111

| 112 | 110 | 111 | 122 | 117 |
|---|---|---|---|---|
| | ↑ i | | | ↑ j |

111 < 112 (i moves right)

122 > 112 (stops at i=3)

117 > 112 (moves left)

122 > 112 (moves left)

111 < 112 (stops j=2)

i > j

swap (112, 111)

| 111 | 110 | 112 | 122 | 117 |
|---|---|---|---|---|

112 is fixed

Step 5:

partion (0-1)

i=1 j=1

i=j

swap (111, 110)

| 110 | 111 |
|---|---|

Sorted

| 110 | 111 | 112 | 122 | 117 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

Step 6: partition (3-4)

i=4 j=4

pivot = 122

i=j so swap (122, 117)

| 117 | 122 |
|---|---|

| 110 | 111 | 112 | 117 | 122 |
|---|---|---|---|---|

Sorted

Step 7: Partition (7-10)

Pivot = 141

i = 8  j = 10

| 141 | 151 | 149 | 147 |
|---|---|---|---|
| 7 | 8 (i) | 9 | 10 (j) |

151 > 141 (i stops at i = 8)

147 > 141 (j moves left)

149 > 141 ( " )

151 > 141 (j stops j = 8)

141 ≥ 141 (stops at j = 7)

i ≥ j

Swap (pivot, A[j])

(141, 141)

sorted

| 141 | 151 | 149 | 147 |
|---|---|---|---|
| 7 | 8 | 9 | 10 |

141 fixed

Step 8: (8-10)

Pivot = 151

i = 9  j = 10

149 < 151 ~~(stop right)~~ ~~(stop)~~

147 < 151 (at i = 10)

147 < 151 (at j = 10)

Swap (151, 147)

| 147 | 149 | 151 |
|---|---|---|

Sorted

| 133 | 141 | 147 | 149 | 151 |
|---|---|---|---|---|

Final sorted array:

| 110 | 111 | 112 | 117 | 122 | 123 | 133 | 141 | 147 | 149 | 151 | 157 |
|---|---|---|---|---|---|---|---|---|---|---|---|

Last element as pivot:

| 157 | 110 | 147 | 122 | 111 | 149 | 151 | 141 | 123 | 112 | 117 | 133 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | pivot |

pivot = A[last (high)]

~~i = first index~~, j = low

j = high - 1

move i right while A[i] < pivot (while)

move j left A[j] > pivot (while)

If i < j swap A[i], A[j]

If i ≥ j swap A[i], pivot

Step1: partition (0-11)

pivot = 133

i = 0, j = 10

157 > 133 (i stops at i=0)

117 < 133 (j stops j=10)

swap (157 ↔ 117)

| 117 | 110 | 147 | 122 | 111 | 149 | 151 | 141 | 123 | 112 | 157 | 133 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | j | | pivot |

110 < 133 (move i right)

147 > 133 (stop at i=2)

157 > 133 (moves left)

112 < 133 (stops at j=9)

swap (147, 112)

| 117 | 110 | 112 | 122 | 111 | 149 | 151 | 141 | 123 | 147 | 157 | 133 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | i | | | | | | | j | | pivot |

112 < 133 (moves i right)

111 < 133 ( " )

149 > 133 (stops at i=5)

147 > 133 (moves left)

123 < 133 (stops at j=8)

swap (149, 123)

| 117 | 110 | 112 | 122 | 111 | 123 | 151 | 141 | 149 | 147 | 157 | 133 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | i | | | | j | | | |

123 < 133 (moves right)
151 > 133 (i stops at i=6)

i > j

Swap (pivot, A[i])

Swap (133, 151)

149 > 133 (moves left)
141 > 133 ( " )
151 > 133 ( " )
123 < 133 (stops at j=5)

| 117 | 110 | 112 | 122 | 111 | 123 | 133 | 141 | 149 | 147 | 157 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|

Pivot 133 fixed

**Step2:** Left (0-5)

Pivot = 123
i = 0 j = 4

| 117 | 110 | 112 | 122 | 111 | 123 |
|---|---|---|---|---|---|

i j pivot

117 < 123 (moves right)
110 < 123 " "
112 < 123 " "
111 < 123 " "
123 = 123 (stops at i=5)

111 < 123 (stays at j=4)

Swap (123, 111)

| 117 | 110 | 112 | 122 | 111 | 123 |
|---|---|---|---|---|---|

Pivot 123 fixed.

**Step3:** Left (0-4)

Pivot = 111
i = 0 j = 3

| 117 | 110 | 112 | 122 | 111 |
|---|---|---|---|---|

i j pivot

117 > 111 (stops)
i = 0

122 > 111 (left)
112 > 111 (moves left)
110 < 111 (stops at j=1)

Swap (117, 110)

| 110 | 117 | 112 | 122 | 111 |
|---|---|---|---|---|

i j

117 > 111 (stops at i=1) 110 < 111 (stops at j=1)

i=j swap(117, 111)

| 110 | 111 | 112 | 122 | 117 |
|---|---|---|---|---|

pivot 111 is fixed

Step 4: (2-4)

pivot = 117

i=2 j=3

~~swap (117, 122)~~

112 < 117 (i moves)

122 > 117 (stops)

j=2 swap (117, 122)

| ~~117~~ | 122 |
|---|---|

| 110 | 111 | 112 | 117 | 122 |
|---|---|---|---|---|

Step 5: (7-11)

| 141 | 149 | 147 | 157 | 151 |
|---|---|---|---|---|
| 7 | 8 | 9 | 10 | 11 |

pivot = 151

i=7 j=10

141 < 151 (i moves right)

149 < 151 ( " )

147 < 151 ( " )

157 > 151 (i stops at i=10)

157 > 151 (j moves left)

147 < 151 (stops)

j=9

i > j

swap (157, 151)

| 141 | 149 | 147 | 151 | 157 |
|---|---|---|---|---|

pivot fixed

Step 6: (7-9)

pivot = 147

| 141 | 149 | 147 |
|---|---|---|
| 7 | 8 | 9 |

i=7 j=8

141 < 147 (moves right)

149 > 147 (at j=8)

149 > 147 (moves left)

141 < 147 (at i=7)

i > j

swap (149, 147)

| 141 | 147 | 149 |
|---|---|---|

Final sorted array:

| 110 | 111 | 112 | 117 | 122 | 123 | 133 | 141 | 147 | 149 | 151 | 157 |
|---|---|---|---|---|---|---|---|---|---|---|---|

(ii) Random element as Pivot elements

choose Random as pivot
Swap pivot with first element
Use the steps used in First element as pivot

157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133

Step 1: 141 = pivot
Swap (pivot, First)

| 141 | 110 | 147 | 122 | 111 | 149 | 151 | 157 | 123 | 112 | 117 | 133 |
|---|---|---|---|---|---|---|---|---|---|---|---|

i stops at 147
j stops at 133
Swap (147, 133)

141, 110, 133, 122, 111, 149, 151, 157, 123, 112, 117, 147

i stops at 149
j stops at 133
Swap (149, 117)

141, 110, 133, 122, 111, 147, 151, 157, 123, 112, 149, 147

i stops at 151
j stops at 112
Swap (151, 112)

141, 110, 133, 122, 111, 147, 112, 157, 123, 151, 149, 147

i stops at 157
j stops at 123
Swap (157, 123)

123, 110, 133, 122, 111, 117, 112, 141, 157, 151, 149, 147

pivot index = 7

Step2: (0-6)

123 | 110 | 133 | 122 | 111 | 117 | 112

Random = 117

| 117 | 110 | 133 | 122 | 111 | 123 | 112 |
|---|---|---|---|---|---|---|

i stops at 1
j stops at 6 Swap (133, 112)

| 117, 110, 112, 122, 111, 123, 133 |
|---|

i stops at 122
j stops at 111 Swap (122, 111)

| 117 | 110 | 112 | 111 | 122 | 123 | 133 |
|---|---|---|---|---|---|---|

i = 4
j = 3 i > j (stop)

Swap (111, 117)

| 111 | 110 | 112 | 117 | 122 | 123 | 133 |
|---|---|---|---|---|---|---|

Step3: (0, 2)

Pivot = 111

Swap with 111

| 110 | 111 | 112 |
|---|---|---|

i = 1, j = 2 i

Already sorted

Right of 117 ⇒ already sorted

Right of 141 ⇒ | 157 | 151 | 149 | 147 |

Step 4: [8–11]

Pivot = 149 | 157 | 151 | 149 | 147

Swap with 157

| 149 | 151 | 157 | 147 |

i = 9

j = 11 Swap (147, 151)

| 149 | 147 | 157 | 151 |

i = 10

j = 9 Swap (149, 147) | 147 | 149 | 157 | 151 |

Step 5: Right of 149 [10, 11]

157 151

Pivot = 151

After swap 151 157

Sorted

Final sorted array:

| 110 | 111 | 112 | 117 | 122 | 123 | 133 | 141 | 147 | 149 | 151 | 157 |