SCHOOL OF COMPUTING

DHEERAJ VAMSI KRISHNA

CH.SC.U4CSE24127

WEEK 5

Design and Analysis of Algorithm(23CSE211)

Quicksort

Code:

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

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

int partition(int arr[], int low, int high, int pivotChoice) {
    int pivotIndex;

    if (pivotChoice == 1) {
        pivotIndex = low;
    } else if (pivotChoice == 2) {
        pivotIndex = high;
    } else {
        pivotIndex = low + rand() % (high - low + 1);
        printf("Random Pivot Element Chosen: %d\n", arr[pivotIndex]);
    }

    swap(&arr[pivotIndex], &arr[high]);

    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high, int pivotChoice) {
    if (low < high) {
        int pi = partition(arr, low, high, pivotChoice);
        quickSort(arr, low, pi - 1, pivotChoice);
        quickSort(arr, pi + 1, high, pivotChoice);
    }
}

int main() {
    printf("CH.SC.U4CSE24127\n");

    int n, pivotChoice;
    printf("Enter number of elements: ");
    if (scanf("%d", &n) != 1) return 1;

    int* arr = (int*)malloc(n * sizeof(int));
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
```

```c
int main() {
    printf("CH.SC.U4CSE24127\n");

    int n, pivotChoice;
    printf("Enter number of elements: ");
    if (scanf("%d", &n) != 1) return 1;

    int* arr = (int*)malloc(n * sizeof(int));
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Choose Pivot Element:\n1. First Element\n2. Last Element\n3. Random Element\nChoice: ");
    scanf("%d", &pivotChoice);

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

    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    free(arr);
    return 0;
}
```

Output:

```
dheeraj@DheerajG15:/mnt/c/Users/medar$ gcc quickflr.c -o quickflr
dheeraj@DheerajG15:/mnt/c/Users/medar$ ./quickflr
CH.SC.U4CSE24127
Enter number of elements: 12
Enter 12 elements: 157 110 147 122 111 149 151 141 123 112 117 133
Choose Pivot Element:
1. First Element
2. Last Element
3. Random Element
Choice: 3
Random Pivot Element Chosen: 141
Random Pivot Element Chosen: 123
Random Pivot Element Chosen: 111
Random Pivot Element Chosen: 117
Random Pivot Element Chosen: 149
Random Pivot Element Chosen: 157
Sorted array: 110 111 112 117 122 123 133 141 147 149 151 157
dheeraj@DheerajG15:/mnt/c/Users/medar$
```

Time Complexity

- **Best Case: O(n log n)**
  When the pivot divides the array into two nearly equal halves.
- **Average Case: O(n log n)**
  Random or good pivot selection gives balanced partitions.
- **Worst Case: O(n^2)**
  When the pivot is always the smallest or largest element (e.g., already sorted array with first/last pivot).

Space Complexity

- **Space Complexity: O(log n)**

Explanation:
Quick Sort uses **recursion**, and no extra arrays are used.
In the average case, the recursion depth is **log n**, so the space used by the recursion stack is **O(log n)**.
(In the worst case, recursion depth can become **O(n)**.)