Name: Ganath Avinash G.R

CSE – B

CH.SC.U4CSE24118

Week –5 (1/2026)

1. Quick Sort : (All pivot selections : First,Last,Random)

Code:

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

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition_first(int arr[], int low, int high) {
    int pivot = arr[low];
    int i = low + 1;

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

```
void quick_first(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition_first(arr, low, high);
        quick_first(arr, low, pi - 1);
        quick_first(arr, pi + 1, high);
    }
}

int partition_last(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;
}

void quick_last(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition_last(arr, low, high);
        quick_last(arr, low, pi - 1);
        quick_last(arr, pi + 1, high);
    }
}
```

```
int partition_random(int arr[], int low, int high) {
    int random_index = low + rand() % (high - low + 1);
    swap(&arr[random_index], &arr[high]);
    return partition_last(arr, low, high);
}

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

int main() {
    int n, choice;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];

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

    printf("\nChoose Pivot Method:\n");
    printf("1. First Element Pivot\n");
    printf("2. Last Element Pivot\n");
```

```
    printf("3. Random Pivot\n");
    printf("Enter choice: ");
    scanf("%d", &choice);

    srand(time(NULL));

    switch(choice) {
        case 1:
            quick_first(arr, 0, n - 1);
            break;
        case 2:
            quick_last(arr, 0, n - 1);
            break;
        case 3:
            quick_random(arr, 0, n - 1);
            break;
        default:
            printf("Invalid choice!\n");
            return 0;
    }

    printf("\nSorted Array:\n");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

OUTPUT:

1st:

```
Enter number of elements: 12
Enter elements:
157 110 147 122 111 149 151 141 123 112 117 133

Choose Pivot Method:
1. First Element Pivot
2. Last Element Pivot
3. Random Pivot
Enter choice: 1

Sorted Array:
110 111 112 117 122 123 133 141 147 149 151 157
```

Last :

```
Enter number of elements: 12
Enter elements:
157 110 147 122 111 149 151 141 123 112 117 133

Choose Pivot Method:
1. First Element Pivot
2. Last Element Pivot
3. Random Pivot
Enter choice: 2

Sorted Array:
110 111 112 117 122 123 133 141 147 149 151 157
```

Random:

```
Enter number of elements: 12
Enter elements:
157 110 147 122 111 149 151 141 123 112 117 133

Choose Pivot Method:
1. First Element Pivot
2. Last Element Pivot
3. Random Pivot
Enter choice: 3

Sorted Array:
110 111 112 117 122 123 133 141 147 149 151 157
```

Complexity:

Space Complexity:O(n)
Time Complexity:O(nlogn)

Best Among All :

Random Pivot is best . Why ?

> Avoids already sorted worst case
> Makes partitions more balanced
> Gives average O(n log n) almost always

Quicksort

Same array (first as pivot)

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

↑ Pivot

S1: 133 110 147 122 111 149 151 141 123 112 117 157

↑ pivot

S2: 133 110 117 122 111 149 151 141 123 112 147 157

↑ pivot

S3: 133 110 117 122 111 112 151 141 123 149 147 157

↑ p

S4: 133 110 117 122 111 112 123 141 151 149 147 157

↑p

S5: 123 110 117 122 111 112 133 141 151 149 147 157

S6: 112 110 117 122 111 123 133 141 151 149 147 157

S7: 112 110 111 122 117 123 133 141 147 149 157

S8: 111 110 112 122 117 123 133 141 147 149 151 157

S9: 110 111 112 117 122 123 133 141 147 149 151 157

=

Last as pivot

S1: 157 110 147 122 111 149 151 141 123 112 117 133

S2: 117 110 147 122 111 149 151 141 123 112 157 133

S3: 117 110 112 122 111 149 151 141 123 147 157 133

S4: 117 110 112 122 111 123 151 141 149 147 157 133

S5: 117 110 112 122 111 123 133 141 149 147 157 252

S2: 117 110 112 122 111 [123] [133] 141 149 147 [151]

157

S7: 110 117 112 122 111 [123 133] 141 147 149 [151] [157]

S8: 110 [111] 112 122 117 [123] [133] 141 [147] 149 [151] [157]

S9: [110] [111] 112 [117] 122 [123] [133] [141] [147] [149] [151] [157]

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

Random:

S1: 157 110 147 122 111 149 151 141 [123] 112 117 133

S2: 117 110 147 122 111 149 151 141 123 112 157 133

S3: 117 110 112 122 111 [123] 151 141 149 147 157 133

S5: 110 117 112 122 111 [123] [133] 141 149 147 157 157

S6: 110 [111] 112 122 117 [123] [133] 141 149 147 157 157

S7: [110] [111] [112] 122 117 [123] [133] 141 149 147 157 157

S8: [110] [111] [112] [117] 122 [123] [133] 141 [147] 149 157 [151]

S9: 110 111 112 117 122 123 133 141 147 149 151 157

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

Algorithm / code: (for int)

```
Partition (L, h) {
    pivot = arr[L];
    i = L ; j = h;

    while (i < j) {
        do {
            i++;
        } while ( Arr[i] ≤ pivot )

        do {
            j--;
        } while (Arr[j] > pivot)

        if (i < j) {
            swap (arr[i], arr[j]);
        }
    } swap (arr[L] , arr[j]);
    return j;
}
```

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

```
Quick_sort (L, h) {
    if ( L < h ) {
        partition_point = partition (L, h);
        Quicksort ( L, partition_point);
        Quick sort (partition_point + 1 , h);
    }
}
```