**DSP LAB**

**Week 7 Assignment**

Performance of Quick-Sort with four different pivots for partitioning

Pivot-1: Choose the first element as pivot

Pivot-2: Choose randomly the pivot element

Pivot-3: Median of {First element, Middle element, Last element} Pivot-4: Median of {n/4th element, middle element, 3n/4th element}

Performance of Merge-Sort

Input Array types:

1. Random numbers of integers in the range 0 to 10k ( k = 4,5,6,7) 2. Sorted array of (1)

3. Almost Sorted of (1). ( This can be obtained by swapping 1% pairs of the  sorted array)

Implement Quick\_Sort(Array, Size\_of\_Array, Pivot\_type)

Implement Merge\_Sort(Array, Size\_of\_Array)

Merge Sort:

Source Code:

#include <stdio.h>

#include <stdlib.h>

#define MAX 1000000

#include<time.h>

long int arr[MAX];

// Merge two subarrays A and B into arr

void merge(long int arr[], int p, int q, int r) {

// Create L = p..q and M = q+1..r

int n1 = q - p + 1;

int n2 = r - q;

int A[n1], B[n2];

for (int i = 0; i < n1; i++)

A[i] = arr[p + i];

for (int j = 0; j < n2; j++)

B[j] = arr[q + 1 + j];

// Maintain current index of sub-arrays and main array

int i, j, k;

i = 0;

j = 0;

k = p;

// elements A and B and place them in the correct position at A[p..r]

while (i < n1 && j < n2) {

if (A[i] <= B[j]) {

arr[k] = A[i];

i++;

} else {

arr[k] = B[j];

j++;

}

k++;

}

// pick up the remaining elements and put in A[p..r]

while (i < n1) {

arr[k] = A[i];

i++;

k++;

}

while (j < n2) {

arr[k] = B[j];

j++;

k++;

}

}

// Divide the array into two subarrays, sort them and merge them

void mergeSort(long int arr[], int l, int r) {

if (l < r) {

// m is the point where the array is divided into two subarrays

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

// Merge the sorted subarrays

merge(arr, l, m, r);

}

}

int main() {

int i, n,choice,k,temp;

while(1)

{

printf("\n 1. Random Integers in the range of 10^k, k between 4 and 7 \n 2. Sorted array \n 3. Almost sorted array \n 4. Exit \n");

printf("\n Enter your Choice:");

scanf("%d", &choice);

if(choice ==1)

{

printf("\n Enter the value of k in 10^k : ");

scanf("%d",&k);

printf("Enter Size of an Array:");

scanf("%d",&n);

int val\_n = pow(10,k);

time\_t t;

printf("\nValues in Array before Sorting:\n");

/\* Intializes random number generator \*/

srand((unsigned) time(&t));

for( i = 0 ; i < n ; i++ ) {

arr[i] = rand() % val\_n;

printf("%ld ", arr[i]);

}

}

else if (choice == 2)

{

printf("\n Enter the value of k in 10^k : ");

scanf("%d",&k);

printf("Enter Size of an Array:");

scanf("%d",&n);

for( i = 0 ; i < n ; i++ ) {

arr[i] = i;

printf("%ld ", arr[i]);

}

}

else if (choice ==3)

{

printf("\n Enter the value of k in 10^k : ");

scanf("%d",&k);

printf("Enter Size of an Array:");

scanf("%d",&n);

for( i = 0 ; i < n ; i++ ) {

arr[i] = i;

int ii = i;

if(i%200==0 && i >0)

{

arr[i] = i - 25;

}

printf("%ld ", arr[i]);

}

}

else if(choice == 4)

{

break;

}

mergeSort(arr, 0, n - 1);

printf("\nSorted array: \n");

for (int i = 0; i < n; i++)

printf("%ld ", arr[i]);

printf("\n");

}

}

Output:







