**Design and implement C Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// Function to merge two subarrays of arr[]

void merge(int arr[], int l, int m, int r) {

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

// Create temporary arrays

int L[n1], R[n2];

// Copy data to temporary arrays L[] and R[]

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

L[i] = arr[l + i];

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

R[j] = arr[m + 1 + j];

// Merge the temporary arrays back into arr[l..r]

i = 0; // Initial index of first subarray

j = 0; // Initial index of second subarray

k = l; // Initial index of merged subarray

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

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

// Copy the remaining elements of L[], if there are any

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

// Copy the remaining elements of R[], if there are any

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

// Merge Sort function

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

if (l < r) {

// Same as (l+r)/2, but avoids overflow for large l and r

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

// Sort first and second halves

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

// Merge the sorted halves

merge(arr, l, m, r);

}

}

int main() {

srand(time(NULL)); // Seed for random number generation

const int numPoints = 10; // Number of data points

const int startingN = 5000; // Starting value of n

const int stepSize = 5000; // Step size for increasing n

printf("n\tTime (seconds)\n");

// Loop through different values of n and record time taken

for (int i = 0; i < numPoints; ++i) {

int n = startingN + i \* stepSize;

int\* arr = (int\*)malloc(n \* sizeof(int));

// Generate random numbers

for (int j = 0; j < n; ++j) {

arr[j] = rand();

}

// Measure time taken for sorting

clock\_t start = clock();

mergeSort(arr, 0, n - 1);

clock\_t end = clock();

double timeTaken = ((double)(end - start)) / CLOCKS\_PER\_SEC;

printf("%d\t%f\n", n, timeTaken);

free(arr); // Free memory allocated for array

}

return 0;

}

Output

n Time (seconds)

5000 0.000827

10000 0.001700

15000 0.002951

20000 0.003962

25000 0.004746

30000 0.005112

35000 0.005756

40000 0.008431

45000 0.008229

50000 0.009897

Plot the graph from the above values