**Design and implement C Program to sort a given set of n integer elements using SelectionSort 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 perform selection sort**

void selectionSort(int arr[], int n) {

for (int i = 0; i < n - 1; ++i) {

int minIndex = i;

for (int j = i + 1; j < n; ++j) {

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

**// Swap the found minimum element with the first element**

int temp = arr[minIndex];

arr[minIndex] = arr[i];

arr[i] = temp;

}

}

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();

selectionSort(arr, n);

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 0.036999

10000 0.145735

15000 0.302012

20000 0.497512

25000 0.779823

30000 1.125217

35000 1.557287

40000 2.031235

45000 2.517561

50000 3.112581

Plot the graph from the above values

**Design and implement C Program to sort a given set of n integer elements using Quick 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 swap two integers**

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

**// Function to partition the array for quick sort**

int partition(int arr[], int low, int 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;

}

**// Quick Sort function**

void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

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();

quickSort(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 0.000671

10000 0.001434

15000 0.002408

20000 0.003347

25000 0.003994

30000 0.005108

35000 0.005864

40000 0.006751

45000 0.007823

50000 0.008461

Plot the graph from the above values