**Student Name:-Ritam Hudait**

**Student Roll No.:- 1905797**

**Algorithm Lab. Class Assignment-6**

**CSE Group 1**

**Date: - 13th August 2021**

1. **Write a program to sort a given set of elements using the Quicksort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted, and 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.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Value of n** | **Quicksort (Time Complexity)** | | |
| **Best case** | **Average case** | **Worst case** |
| **1** | **5000** | **0.000700** | **0.057448** | **0.087979** |
| **2** | **10000** | **0.001461** | **0.228433** | **0.339017** |
| **3** | **15000** | **0.002196** | **0.210823** | **0.764959** |
| **4** | **20000** | **0.002992** | **0.909749** | **1.389208** |
| **5** | **25000** | **0.003935** | **1.459238** | **2.123645** |
| **6** | **30000** | **0.004785** | **2.019497** | **3.070795** |
| **7** | **35000** | **0.005742** | **2.818689** | **4.125593** |
| **8** | **40000** | **0.006110** | **3.637601** | **5.445079** |
| **9** | **45000** | **0.007339** | **4.612464** | **6.753767** |
| **10** | **50000** | **0.008241** | **5.554096** | **8.367727** |

**Program**

**#include <stdio.h>**

**#include <time.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 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)**

**{**

**if (low < high)**

**{**

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

**quickSort(arr, low, pi - 1);**

**quickSort(arr, pi + 1, high);**

**}**

**}**

**int main() {**

**printf("n\t\t|\tworst\t\tavg\t\t\tbest\n\_\_|\_\_\_\_\n");**

**int sizes;**

**scanf("%d",&sizes);**

**for(int i=0;i<sizes;i++) {**

**int n;**

**scanf("%d", &n);**

**printf("%d\t|\t", n);**

**int arr[n];**

**time\_t start, end;**

**double time;**

**// Worst**

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

**arr[j] = j + 1;**

**}**

**start = clock();**

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

**end = clock();**

**time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;**

**printf("%f\t", time);**

**// Avg**

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

**arr[j] = n - j;**

**}**

**start = clock();**

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

**end = clock();**

**time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;**

**printf("%f\t", time);**

**// Best**

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

**arr[j] = rand() % 10000;**

**}**

**start = clock();**

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

**end = clock();**

**time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;**

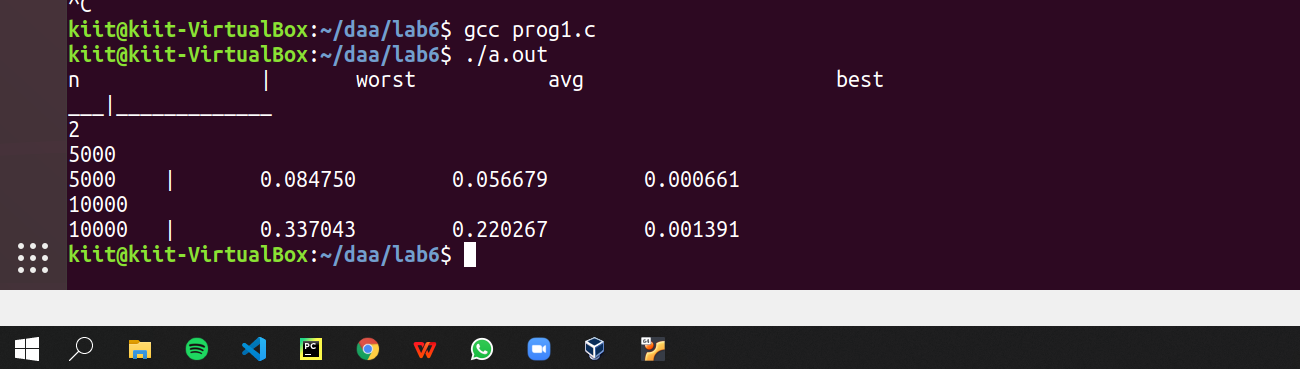
**printf("%f\n", time);**

**}**

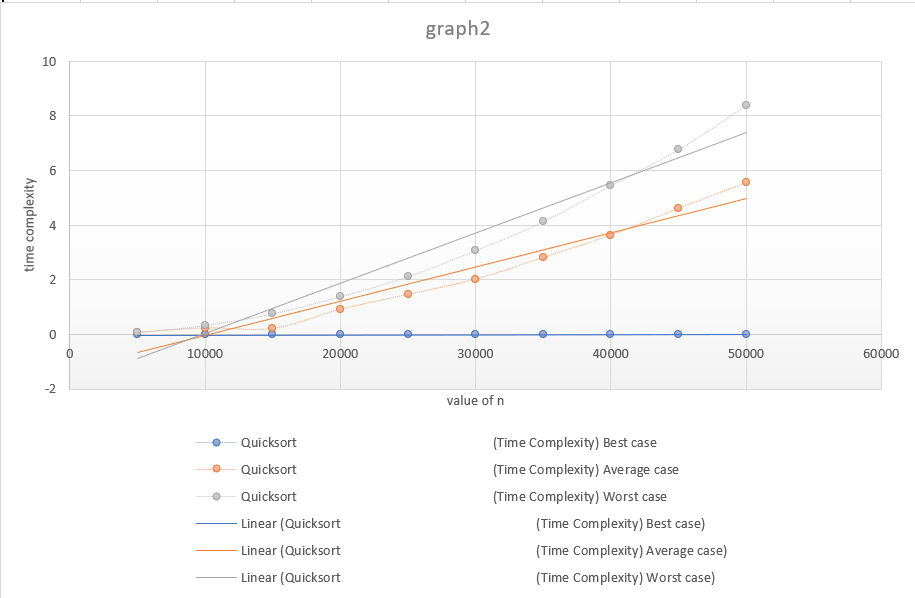
**return 0;**

**}**

**Output**

****

**Graph**

****

1. **Write a program to use the divide and conquer method to recursively implement and to find the maximum and minimum in a given list of n elements.**

|  |  |
| --- | --- |
| **SL No.** | **Input (n)** |
| **(Time complexity)** |
| **1** | **5000** | **0.000227** |
| **2** | **10000** | **0.000391** |
| **3** | **15000** | **0.000498** |
| **4** | **20000** | **0.001207** |
| **5** | **25000** | **0.000759** |
| **6** | **30000** | **0.001988** |
| **7** | **35000** | **0.001263** |
| **8** | **40000** | **0.001175** |
| **9** | **45000** | **0.001753** |
| **10** | **50000** | **0.001492** |

**Program**

**#include <stdio.h>**

**#include <time.h>**

**#include <stdlib.h>**

**int find\_max(int arr[], int s, int e) {**

**if (s > e) {**

**return -(1 << 30);**

**}**

**if (s == e) {**

**return arr[s];**

**}**

**int rem = find\_max(arr, s + 1, e);**

**return ((arr[s] > rem) ? arr[s] : rem);**

**}**

**int find\_min(int arr[], int s, int e) {**

**if (s > e) {**

**return -(1 << 30);**

**}**

**if (s == e) {**

**return arr[s];**

**}**

**int rem = find\_max(arr, s + 1, e);**

**return ((arr[s] < rem) ? arr[s] : rem);**

**}**

**int main() {**

**printf("n\t\t|\tTime\t\tMax\t\tMin\n");**

**int sizes;**

**scanf("%d",&sizes);**

**for(int i=0;i<sizes;i++) {**

**int n;**

**scanf("%d",&n);**

**printf("%d\t|\t", n);**

**int arr[n];**

**time\_t start, end;**

**double time;**

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

**arr[j] = rand() % 5000 + 1;**

**}**

**start = clock();**

**int ans1 = find\_max(arr, 0, n - 1);**

**int ans2 = find\_min(arr, 0, n - 1);**

**end = clock();**

**time = (end - start) \* 1.0 / CLOCKS\_PER\_SEC;**

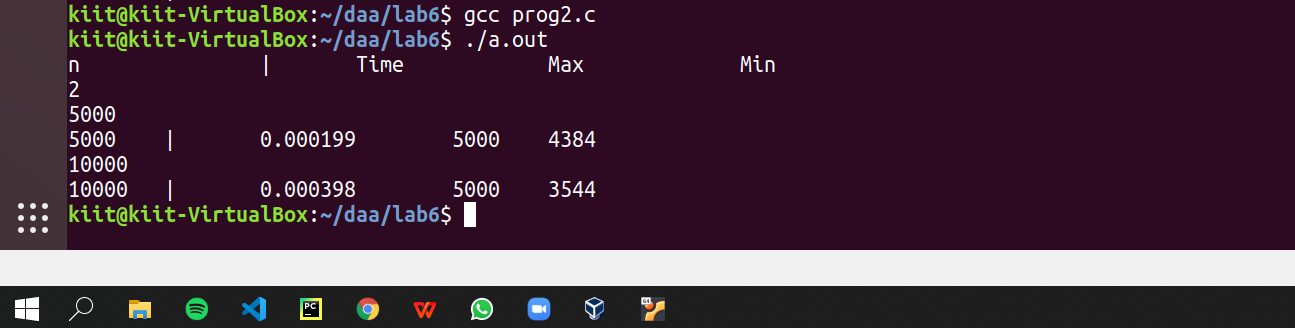
**printf("%f\t%d\t%d\n", time, ans1, ans2);**

**}**

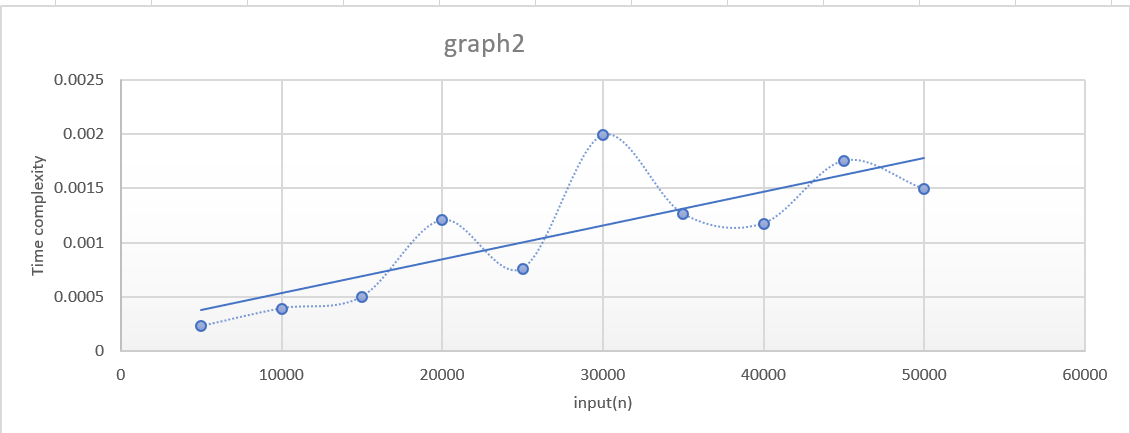
**return 0;**

**}**

**Output**

****

**Graph**

****