**Student Name:-Ritam Hudait**

**Student Roll No.:- 1905797**

**Algorithm Lab. Class Assignment-9**

**CSE Group 1**

**Date: - 24th Sept. 2021**

1. **Write a program to find the kth minimum and maximum element in Heap.**

**Program**

**#include <stdio.h>**

**#include <time.h>**

**#include <stdlib.h>**

**#define pfs(x) printf("%d ", x)**

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

**{**

**int t = \*a;**

**\*a = \*b;**

**\*b = t;**

**}**

**void heapify(int \*arr, int n, int i)**

**{**

**int largest = i;**

**int l = 2 \* i + 1;**

**int r = 2 \* i + 2;**

**if (l < n && arr[l] > arr[largest])**

**largest = l;**

**if (r < n && arr[r] > arr[largest])**

**largest = r;**

**if (largest != i) {**

**swap(arr + i, arr + largest);**

**heapify(arr, n, largest);**

**}**

**}**

**void buildHeap(int \*arr, int n)**

**{**

**int startIdx = (n / 2) - 1;**

**for (int i = startIdx; i >= 0; i--) {**

**heapify(arr, n, i);**

**}**

**}**

**int extract\_maximum(int \*arr, int n) {**

**int m = \*arr;**

**arr[0] = arr[n - 1];**

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

**return m;**

**}**

**int kthMax(int \*arr, int n, int k) {**

**int ans = 0;**

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

**ans = extract\_maximum(arr, n);**

**n--;**

**}**

**return ans;**

**}**

**int main() {**

**printf("n\t\t|\tElement |\tTime\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];**

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

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

**}**

**buildHeap(arr, n);**

**time\_t start, end;**

**double time;**

**// Time**

**start = clock();**

**pfs(kthMax(arr, n, 5));**

**printf("\t|\t");**

**end = clock();**

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

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

**}**

**return 0;**

**}**

**Output**

****

1. **Write a program to recursively implement Binary Search using divide and conquer method. Determine the time required to search an element in an array of n integers. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n. The n integers can be generated randomly.**

**Program**

**#include <stdio.h>**

**#include <time.h>**

**#include <stdlib.h>**

**#define pfs(x) printf("%d ", x)**

**int binSearch(int \*a, int s, int e, int x) {**

**if (s > e) {**

**return -1;**

**}**

**int m = (s + e) / 2;**

**if (a[m] == x)**

**return m;**

**if (a[m] > x)**

**return binSearch(a, s, m - 1, x);**

**if (a[m] < x)**

**return binSearch(a, m + 1, e, x);**

**}**

**int main() {**

**printf("n\t\t|\tworst\t\t\tavg\t\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];**

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

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

**}**

**time\_t start, end;**

**double time;**

**// Worst**

**start = clock();**

**pfs(binSearch(arr, 0, n - 1, 1));**

**end = clock();**

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

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

**// Avg**

**start = clock();**

**pfs(binSearch(arr, 0, n - 1, 1000));**

**end = clock();**

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

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

**// Best**

**start = clock();**

**pfs(binSearch(arr, 0, n - 1, (n - 1) / 2));**

**end = clock();**

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

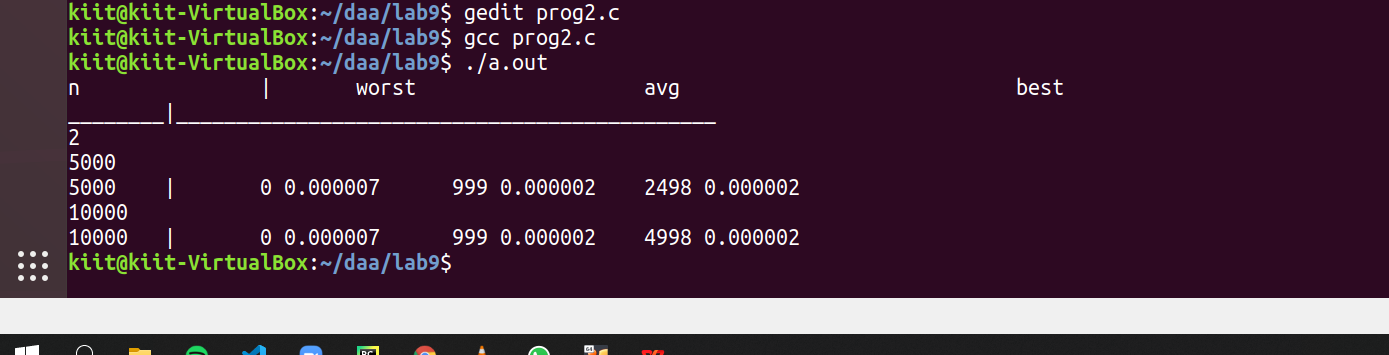
**printf("%f\n", time);**

**}**

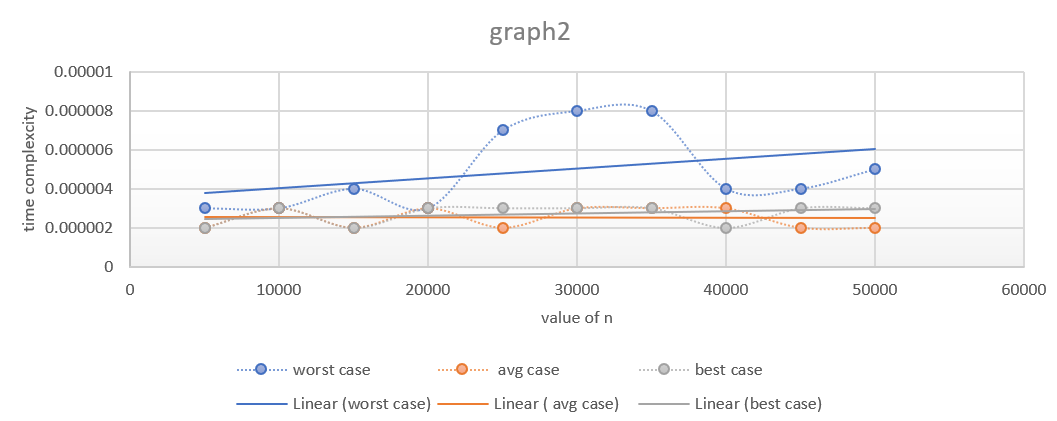
**return 0;**

**}**

**Output**

****

**Graph**

****

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

**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;**

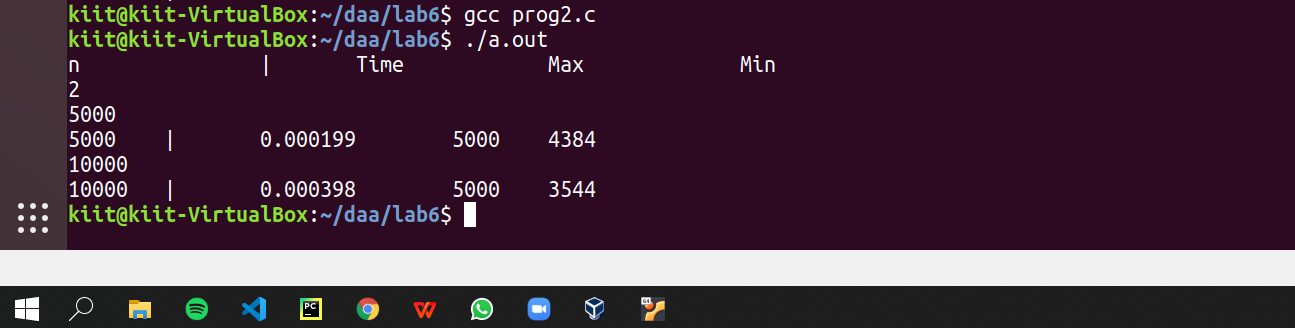
**printf("%f\t%d\t%d\n", time, ans1, ans2);**

**}**

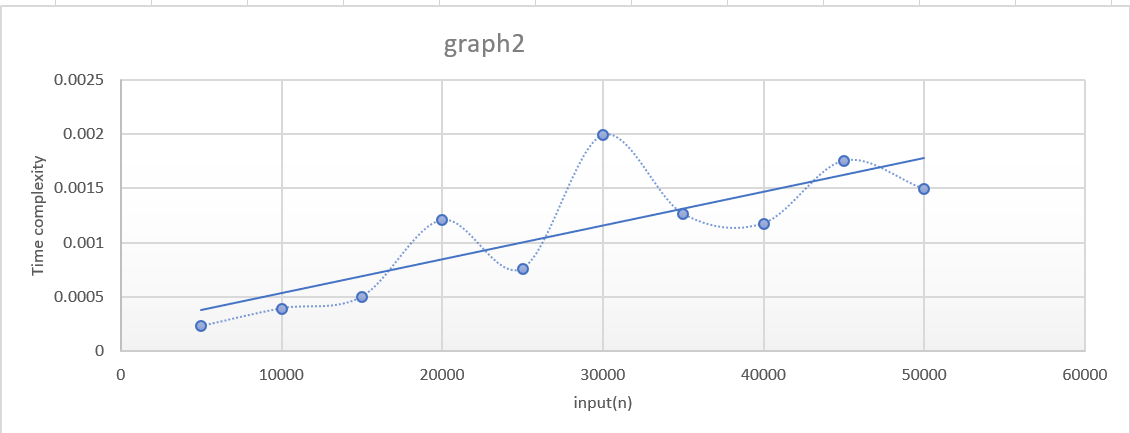
**return 0;**

**}**

**Output**

****

**Graph**

****