DAA – Practical 1

Name: Aditya.Rajesh.Wanwade

Roll no :27 && Batch:C2

Aim:

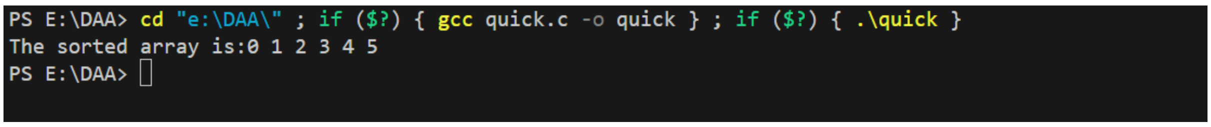
a. Sort a given set by of n integer element using quick sort method and compute its time complexity. Run the program for varied values of n > 5000 and record the time taken to sort the elements can be read from file or can be generated using the random numbers generator. Demonstrate how the divide and conquer method works along with its time complexity analysis: worst case, average case, and best case.

Code of quick sort :

```
#include<stdio.h>
#include<stdlib.h>
struct array{
int size;
int* arr;
};

void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}


int partition(struct array*s, int low, int high) {


  int pivot = s->arr[high];


  int i = (low - 1);

  for (int j = low; j < high; j++) {
    if (s->arr[j] <= pivot) {


      i++;

      swap(&s->arr[i], &s->arr[j]);
```

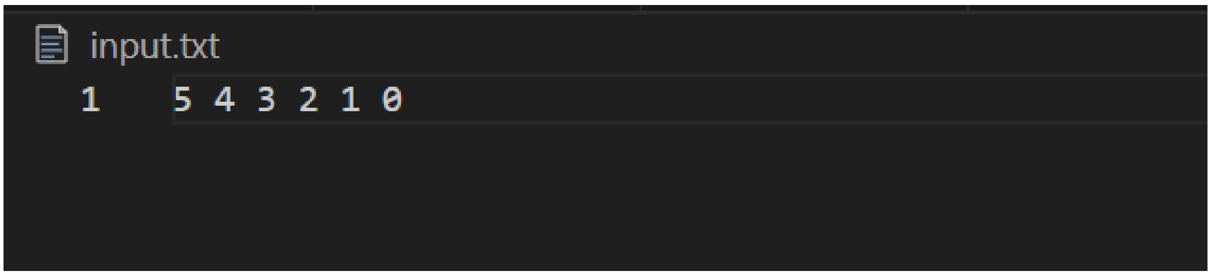
```c
    }
  }


  swap(&s->arr[i + 1], &s->arr[high]);


  return (i + 1);
}

void quickSort(struct array*s, int low, int high) {
  if (low < high) {


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


    quickSort(s, low, pi - 1);


    quickSort(s, pi + 1, high);
  }
}


void printArray(struct array* s) {
  for (int i = 0; i < s->size; i++) {
    printf("%d ", s->arr[i]);
  }
  printf("\n");
}




int main(){
    int i = 0;
    struct array *s = (struct array *)malloc(sizeof(struct array));


    FILE *file = fopen("input.txt", "r");

    if (file == NULL) {
        fprintf(stderr, "Error opening the file.\n");
        return 1;
    }


    fscanf(file, "%d", &(s->size));
    s->arr = (int *)malloc(s->size * sizeof(int));


    for (i = 0; i < s->size; i++) {
        fscanf(file, "%d", &(s->arr[i]));
    }


    fclose(file);


    quickSort(s, 0, s->size-1);


    printf("The sorted array is:");
    printArray(s);


    free(s->arr);
    free(s);

    return 0;
}
```

Output:



```
PS E:\DAA> cd "e:\DAA\" ; if ($?) { gcc quick.c -o quick } ; if ($?) { .\quick }
The sorted array is:0 1 2 3 4 5
PS E:\DAA> 
```

Screenshot of input.txt file:



```
input.txt
5 4 3 2 1 0
```

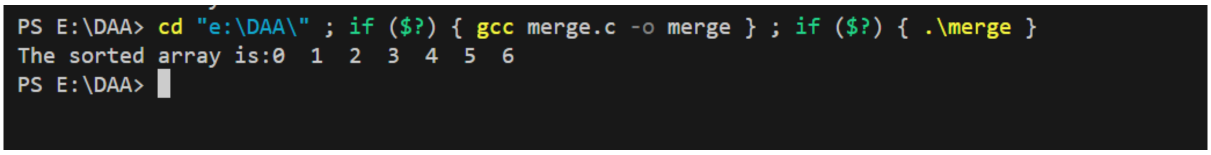
b. Sort a given set of n integer element using merge sort method and compute its time complexity. Run the program from varied values of n > 5000, and record the time taken to sort the elements can read from a file or can be generated using the random number generator. Demonstrate how the divide and conquer method works along with its time complexity analysis: worst case, average case and best case.

Code of merge sort:

```
#include<stdio.h>
#include<stdlib.h>
struct array{
    int size;
    int*arr;
};


void merge(struct array*s, int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;


    int t[n1], t1[n2];


    for (i = 0; i < n1; i++)
        t[i] = s->arr[l + i];
    for (j = 0; j < n2; j++)
        t1[j] = s->arr[m + 1 + j];


    i = 0;


    j = 0;

    k = l;
    while (i < n1 && j < n2) {
        if (t[i] <= t1[j]) {
```

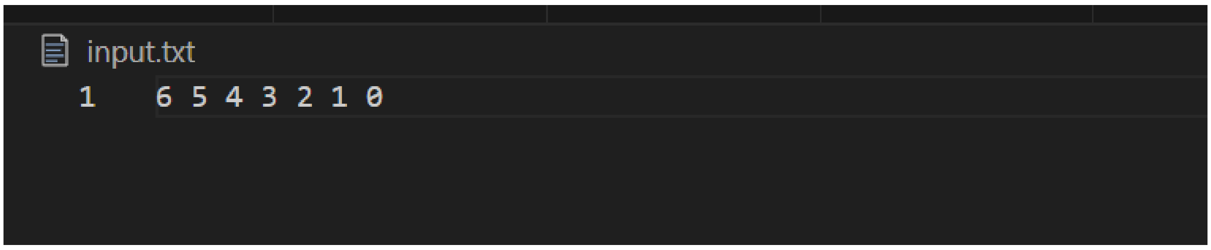
```c
            s->arr[k] = t[i];
            i++;
        }
        else {
            s->arr[k] = t1[j];
            j++;
        }
        k++;
    }


    while (i < n1) {
        s->arr[k] = t[i];
        i++;
        k++;
    }


    while (j < n2) {
        s->arr[k] = t1[j];
        j++;
        k++;
    }
}


void mergeSort(struct array*s, int l, int r)
{
    if (l < r) {

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


        mergeSort(s, l, m);
        mergeSort(s, m + 1, r);

        merge(s, l, m, r);
    }
}

void printArray(struct array* s) {
  for (int i = 0; i < s->size; i++) {
    printf("%d  ", s->arr[i]);
  }
  printf("\n");
}


int main(){
 int i = 0;
    struct array *s = (struct array *)malloc(sizeof(struct array));


    FILE *file = fopen("input.txt", "r");

    if (file == NULL) {
        fprintf(stderr, "Error opening the file.\n");
        return 1;
    }


    fscanf(file, "%d", &(s->size));


    s->arr = (int *)malloc(s->size * sizeof(int));


    for (i = 0; i < s->size; i++) {
        fscanf(file, "%d", &(s->arr[i]));
    }


    fclose(file);


    mergeSort(s, 0, s->size-1);


    printf("The sorted array is:");
    printArray(s);


    free(s->arr);
    free(s);

    return 0;

}
```

Output :



```
PS E:\DAA> cd "e:\DAA\" ; if ($?) { gcc merge.c -o merge } ; if ($?) { .\merge }
The sorted array is:0  1  2  3  4  5  6
PS E:\DAA> 
```

Screenshot of input.txt file:



input.txt

```
6 5 4 3 2 1 0
```