KIET Group of Institutions, Ghaziabad

Department of Computer Applications

(An ISO – 9001: 2015 Certified & 'A' Grade accredited Institution by NAAC)

Design and Analysis of Algorithm

RCA 352: Session 2020-21

DAA Lab

Experiment-No.8

Objective: Implement the **Heap_ sort** algorithm to sort the given list of N numbers and plot graph

Scheduled Date:	Compiled Date:	Submitted Date:
22-9-2020	22-9-2020	22-9-2020

Algorithm:

MaxHeapify(A, i)
```
 1. l  left(i)
 2. r  right(i)
 3. if l  heap-size[A] and A[l] > A[i]
 4. then largest  l
 5. else largest  i
 6. if r  heap-size[A] and A[r] > A[largest]
 7. then largest  r
 8. if largest i
 9. then exchange A[i] and A[largest]
10. MaxHeapify(A, largest)
```

BuildMaxHeap(A)
```
1. heap-size[A]  length[A]
2. for i  length[A]/2 downto 1
3. do MaxHeapify(A, i)
```

HeapSort(A)
```
1. Build-Max-Heap(A)
2. for i  length[A] downto 2
3. do exchange A[1] and A[i]
4. heap-size[A]  heap-size[A] – 1
5. MaxHeapify(A, 1)
```

```
Program file heap_sort.c :
#include <stdio.h>
#include <stdlib.h>
static int count=0;
int main()
{
    int a[50],n;
    void heap_sort(int a[],int);
    void bulid_heap(int [],int);
    void heapify(int [],int,int);
    void get_data(int [],int);
    void put_data(int [],int);
    printf("Enter the size of an array should be less than 50:\n");
    scanf("%d",&n);
    printf("Enter the array elements:\n");
    get_data(a,n);
    printf("\n array before sorting\n");
    put_data(a,n);
    heap_sort(a,n);
    printf("\n array after sorting\n");
    put_data(a,n);
    printf("\n for n=%d counts are %d",n,count);


    return 0;
}
void swap(int *a,int *b)
{
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;
}
void heapify(int a[],int n,int i)
{
    count++;
    int largest=i;
    count++;
    int left=2*i+1;
    count++;
    int right=2*i+2;
    count++;
    if(left < n && a[left]>a[largest])
    {
        count++;
        largest=left;
        count++;
    }
    if(right < n && a[right]>a[largest])
    {
        count++;
        largest=right;
        count++;
    }
    if(largest!=i)
    {
        count++;
        swap(&a[i],&a[largest]);
        count++;
        heapify(a,n,largest);
        count++;
    }

}
void build_heap(int a[],int n)
{
    count++;
    int i;
    count++;
    for(i=(n/2)-1;i>=0;i--)
    {
        count++;
        count++;
        heapify(a,n,i);
        count++;
    }
}
void heap_sort(int a[],int n)
{
    int i;
    count++;
    build_heap(a,n);
    count++;
    for(i=n-1;i>=0;i--)
    {
        count++;
        count++;
        swap(&a[0],&a[i]);
        count++;
        heapify(a,i,0);
        count++;
    }

}
void get_data(int a[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }

}
void put_data(int a[],int n)
```

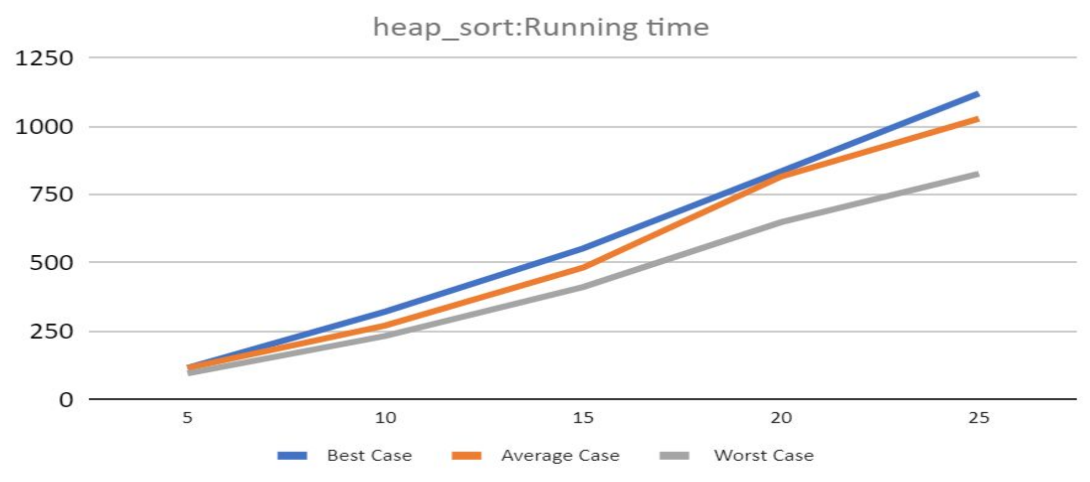
```
{
     int i;
     for(i=0;i<n;i++)
     {
          printf("%d\t",a[i]);
     }

}
```

Output

Input Size	Best Case	Average Case	Worst Case
5	116	116	96
10	322	271	233
15	553	483	412
20	835	816	649
25	1120	1028	826

Graph



Conclusion

KIET Group of Institutions, Ghaziabad

Department of Computer Applications

(An ISO – 9001: 2015 Certified & 'A' Grade accredited Institution by NAAC)

Design and Analysis of Algorithm

RCA 352: Session 2020-21

DAA Lab

Case	Running Time : Growth of Function mathematically	Running Time : Growth of Function after observing graph
Best Case	O($nlogn$)	O($nlogn$)
Average Case	O($nlogn$)	O($nlogn$)
Worst Case	O($nlogn$)	O($nlogn$)