Ayush Goyal

190905522 CSE D 62

DAA Lab 8 (Week 8) - TRANSFORM AND CONQUER – II

1. **Write a program to create a heap for the list of integers using top-down heap construction algorithm and analyse its time efficiency. Obtain the experimental results for order of growth and plot the result.**

CODE:

```c
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

int op=0;

void heapify(int arr[], int currIndex)
{
    int parent = (currIndex)/2;
    op++;
    while(parent > 0)
    {
        op++;
        if(arr[parent]<arr[currIndex])
        {
            int temp = arr[parent];
            arr[parent] = arr[currIndex];
            arr[currIndex] = temp;

            currIndex = parent;
            parent = (currIndex)/2;
        }
        else
            return;
    }
}


int main(){
    int n;
    printf("\nEnter number of elements : ");
    scanf("%d",&n);
    int *h = (int*)malloc((sizeof(int))*(n+1));
    printf("\nEnter elements : ");
    for(int i=1;i<=n;i++){
        scanf("%d",&h[i]);
        heapify(h,i);
```

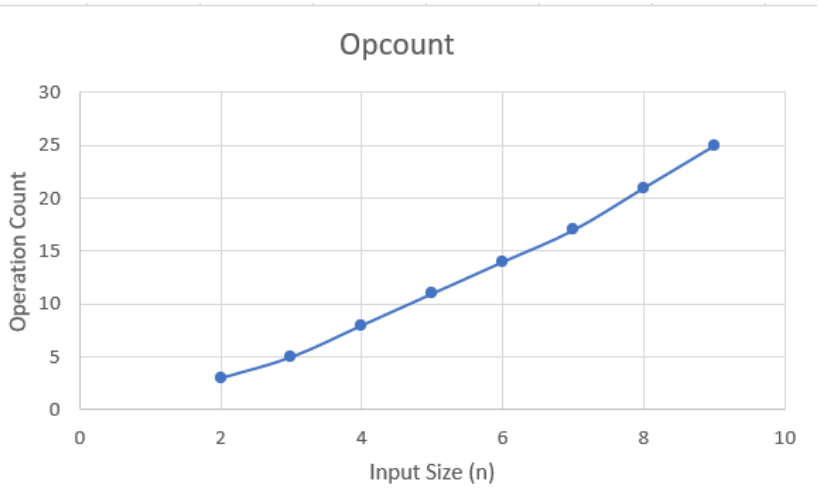
```
    }
    printf("\nThe heap created is : ");
    for(int i=1;i<=n;i++){
        printf("%d ",h[i]);
    }
    printf("\n");
    printf("\nThe Opcount is : %d",op);
    return 0;
}
```

OUTPUT:

```
D:\CSE\CSE Labs\DAA Lab\Week 8 - Transform and Conquer - 2>gcc topdownheap.c -o topdown

D:\CSE\CSE Labs\DAA Lab\Week 8 - Transform and Conquer - 2>topdown

Enter number of elements : 6

Enter elements : 2 9 7 6 5 8

The heap created is : 9 6 8 2 5 7

The Opcount is : 13
D:\CSE\CSE Labs\DAA Lab\Week 8 - Transform and Conquer - 2>
```

Graph and Table Plot:

Input size(n)	Opcount
2	3
3	5
4	8
5	11
6	14
7	17
8	21
9	25



Time Efficiency Analysis:

The input is an array of ascending order, and the size is n. We can see that the operation count is close to n*log n. Therefore it is of the order of growth of **O(n*log n).**
Basic Operation is Comparison.

2. **Write a program to sort the list of integers using heap sort with bottom up max heap construction and analyse its time efficiency. Prove experimentally that the worst case time complexity is O(nlog n)**

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

int op=0;

void heapify(int *h, int n){
    int i,j,k,v;
    bool heap;
    for(i=(n/2);i>=1;i--){
        k=i;
        v=h[k];
        heap = false;
        while(!heap && (2*k) <= n){
            op++;
            j=2*k;
            if(j<n)
                if(h[j]<h[j+1])
                    j=j+1;
            if(v>=h[j])
                heap=true;
            else{
                h[k]=h[j];
                k=j;
            }
        }
        h[k]=v;
    }
    return;
}

void heapsort(int *h, int n){
    int i,k=n,temp;
    fflush(stdin);
    for(i=1;i<n;i++){
        op++;
        heapify(h,k);
        temp = h[1];
        h[1] = h[k];
        h[k] = temp;
        k=k-1;
    }
}
```

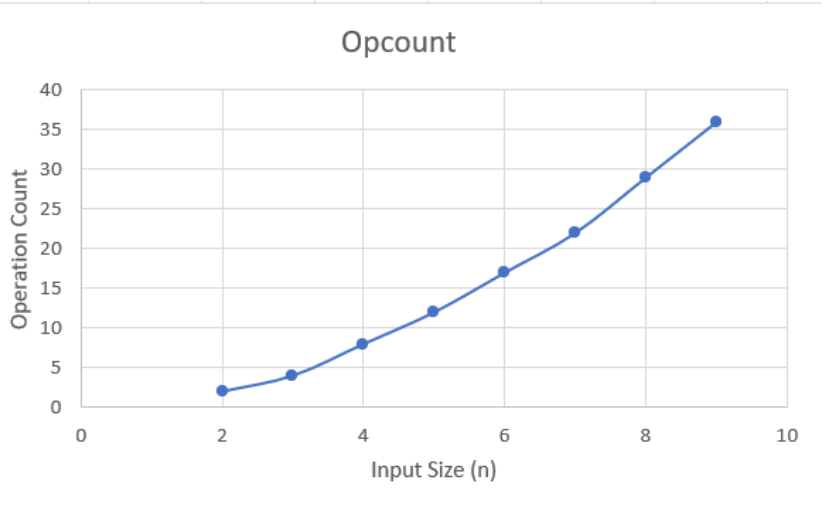
```c
int main(){
    int n;
    printf("\nEnter number of elements : ");
    scanf("%d",&n);
    int *h = (int*)malloc((sizeof(int))*(n+1));
    printf("\nEnter elements : ");
    for(int i=1;i<=n;i++){
        scanf("%d",&h[i]);
    }
    heapsort(h,n);
    printf("\nThe sorted array created is : ");
    for(int i=1;i<=n;i++){
        printf("%d ",h[i]);
    }
    printf("\n");
    printf("\nThe Opcount is : %d",op);
    return 0;
}
```

OUTPUT:

```
D:\CSE\CSE Labs\DAA Lab\Week 8 - Transform and Conquer - 2>gcc heapsort.c -o heapsort

D:\CSE\CSE Labs\DAA Lab\Week 8 - Transform and Conquer - 2>heapsort

Enter number of elements : 6

Enter elements : 2 9 7 6 5 8

The sorted array created is : 2 5 6 7 8 9

The Opcount is : 15
D:\CSE\CSE Labs\DAA Lab\Week 8 - Transform and Conquer - 2>
```

Graph and Table Plot:

Input size(n)	Opcount
2	2
3	4
4	8
5	12
6	17
7	22
8	29
9	36



Time Efficiency Analysis:

The input is an array of ascending order, and the size is n, so we can see O(nlogn) for worst case. We can see that the operation count is close to n*log n. Therefore it is of the order of growth of **O(n*log n).**
Basic Operation is Comparison.

THE END