20/6/24 Week-8

→ heapsort

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
void heap_sort(int a[], int n);
void heapify(int a[], int n, int i);
void print_array(int a[], int n);
void main(){
    int a[15000], n, j, i, ch;
    clock_t start, end;
    while(1){
        printf("\n 1: For manual entry of N value and array
                    elements");
        printf("\n 2: For To display time taken for sorting
            number of elements N in the range 500 to 14500");
        printf("\n 3: To exit");
        printf("\n Enter your choice:");
        scanf("%d", &ch);
        switch(ch){
            case 1:
                printf("\n Enter the number of elements:");
                scanf("%d", &n);
                printf("\n Enter array elements:");
                for(i=0; i<n; i++){
                    scanf("%d", &a[i]);
                }
                start = clock();
                heap_sort(a,n);
                end= clock();
                printf("\n Sorted array is:");
                print_array(a, n);
                printf("\n Time taken to sort %d numbers is
```

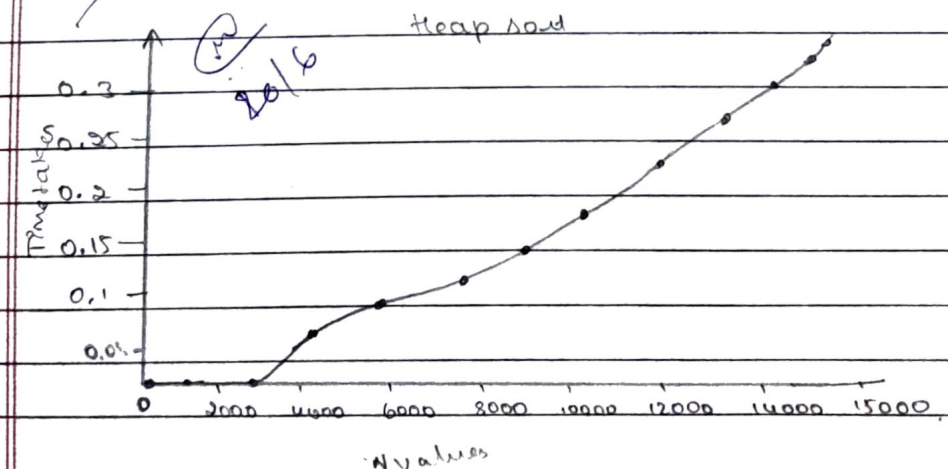
```
%f secs",n,(((double)(end-start))/clocks_per_sec));
break;
case 2:
n=500;
while(n<=11500){
    for(i=0; i<n; i++){
        a[i]=n-i;
    }
    start=clock();
    heap_sort(a,n);
    for(i=0; i<5000000; i++){
        int temp=38/600;
    }
    end=clock();
    printf("\nTime taken to sort %d numbers is %f
        secs",n,(((double)(end-start))/clocks_per_sec));
    break;n+=1000;
    }
    break;
    case 3:
        exit(0);
    }
    getchar();
  }
}
void heap_sort(int a[], int n){
    for(int i=n/2-1; i>=0; i--){
        heapify(a,n,i);
    }
    for(int i=n-1; i>0; i--){
        int temp=a[0];
        a[0]=a[i]
```

```
        a[i]=temp;
        heapify(a,i,0);
      }
    }
void heapify(int a[], int n, int i){
    int largest=i;
    int left=2*i+1;
    int right=2*i+2;
    if(left<n && a[left]>a[largest]){
        largest=left;
    }
    if(right<n && a[right]>a[largest]){
        largest=right;
    }
    if(largest!=i){
        int temp=a[i];
        a[i]=a[largest];
        a[largest]=temp;
        heapify(a,n,largest);
      }
    }
void print array(int a[], int n){
    for(int i=0; i<n; i++){
        printf("%d\t", a[i]);
      }
    }
```

output:

1. For manual entry of N value and array elements
2. To display time taken for sorting number of elements N in the range 500 to 14500
3. To exit.

Enter your choice: 1.
Enter yout the number of elements: 8
Enter array elements: 10 20 15 20 25 35 50 60
Sorted array is: 10 15 20 20 25 35 50 60.
1. For manual entry of N value and array elements
2. To display time taken for sorting number of elements
N in the range 500 to 14500
3. To exit.
Enter your choice: 2
Time taken to sort 500 numbers is 0.031000 secs.
Time taken to sort 1500 numbers is 0.016000 secs
Time taken to sort 2500 numbers is 0.015000 secs
Time taken to sort 3500 numbers is 0.016000 secs
Time taken to sort 4500 numbers is 0.031000 secs
Time taken to sort 5500 numbers is 0.016000 secs
Time taken to sort 6500 numbers is 0.016000 secs
Time taken to sort 7500 numbers is 0.031000 secs
Time taken to sort 8500 numbers is 0.015000 secs
Time taken to sort 9500 numbers is 0.016000 secs
Time taken to sort 10500 numbers is 0.016000 secs
Time taken to sort 11500 numbers is 0.031000 secs
Time taken to sort 12500 numbers is 0.016000 secs
Time taken to sort 13500 numbers is 0.015000 secs
Time taken to sort 14500 numbers is 0.031000 secs.



```
// Floydwarshall
#include <stdio.h>
#define V 5
#define INF 99999
void printsolution(int dist[][V]);
void floydwarshall(int dist[][V])
{
    int i, j, k;
    for(k=0; k<V; k++){
        for(i=0; i<V; i++){
            for(j=0; j<V; j++){
                if(dist[i][k]+dist[k][j]<dist[i][j])
                dist[i][j]=dist[i][k]+dist[k][j];
                }
            }
        }
    printsolution(dist);
}
void printsolution(int dist[][V])
{
    printf("The following matrix shows the shortest distances"
        "between every pair of vertices\n");
    for(int i=0; i<V; i++){
        for(int j=0; j<V; j++){
            if(dist[i][j]==INF)
                printf("%7s", "INF");
            else
                printf("%7d", dist[i][j]);
        }
        printf("\n");
        }
    }
```

```
int main()
{
    int graph[V][V] = { {0, 4, INF, 5, INF},
                        {INF, 0, 1, INF, 6},
                        {2, INF, 0, 3, INF},
                        {INF, INF, 1, 0, 2},
                        {1, INF, INF, 4, 0} };
    floydWarshall(graph);
    return 0;
}
```

Output:

The following matrix shows the shortest distances between every pair of vertices.

```
0   4   5   5   7
3   0   1   4   6
2   6   0   3   5
3   7   1   0   2
1   5   5   4   0
```

20/6/24