Floyd Warshall

```
#include <stdio.h>
#define V 4
#define INF 9999

void printSolution(int dist[][V]);

void printsolu 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])
{
    pf("The foll. matrix shows the shortest
    dibances btw every pair of vertices
    \n");
    for (int i=0; i<V; i++) {
        for (int j=0; j<V; j++) {
            if (dist[i][j] == INF)
                pf("%7s", "INF");
            else
```

```
        pf("%7d", dist[i][j]);
    }
    pf("\n");
  }
}

int main()
{
  int graph[V][V] = { {0, 5, INF, 10},
                      {INF, 0, 3, INF},
                      {INF, INF, 0, 1},
                      {INF, INF, INF, 0} };

  floydWarshall(graph);
  return 0;
}
```

Output:-

The following matrix shows the shortest distance btw every pair of vertices.

```
  0    5    8    9
INF    0    3    4
INF  INF    0    1
INF  INF  INF    0
```

Heap Sort

```
#include <time.h>

void bottomupheapify(int N, int a[], int i)
    int f
        int lar = i;
        int left = 2*i+1
        int right = 2* i+2;
    if (left<N && arr[left] > arr[largest])
            largest = left;
    if (right<N && arr[right] > arr[largest])
            largest = right;

    if (largest != i)
            swap(&arr[i], &arr[largest]);
        heapify (arr, N, largest);
    }
}
void heapsort (int arr[], int N)
{
    for (int i = N/2 -1 ; i>=0 ; i--)
            heapify (arr,N, i);
    for(int i=N-1 ; i>=0 ; i--) {
        swap(&arr[0], &arr[i]);
        heapify(arr, i, 0);
    }
}

void printArray(int arr[], int N)
{   for (int i=0 ; i<N; i++)
            pf("%d", arr[i]);
        pf("\n");
```

```
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int N = sizeof(arr) / sizeof(arr[0]);
    heapSort(arr, N);
    printf("Sorted array is \n");
    printArray(arr, N);
}
```

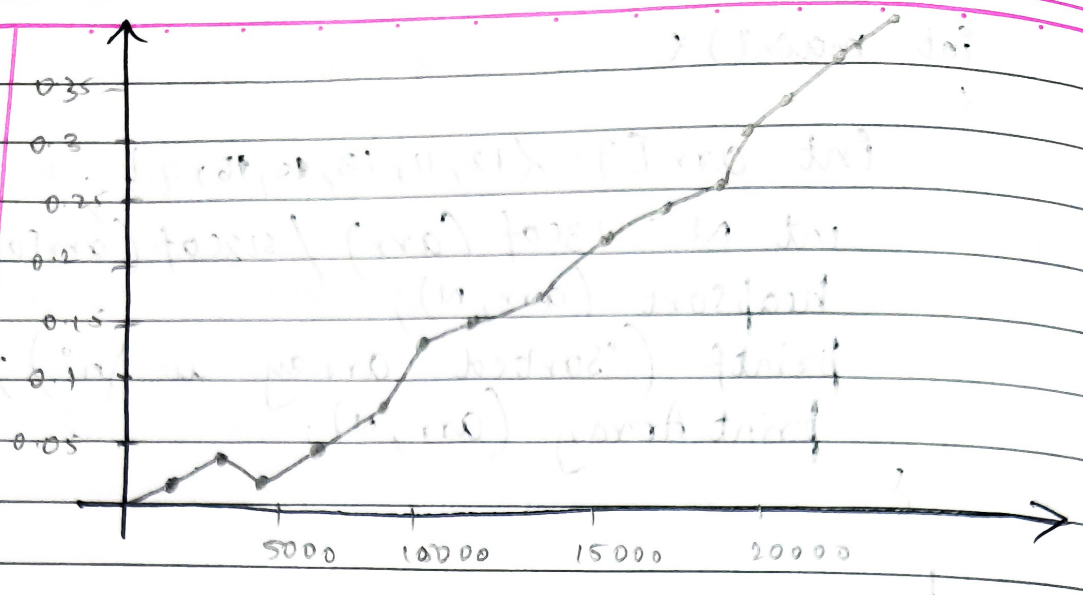
O/P:

Sorted array is:

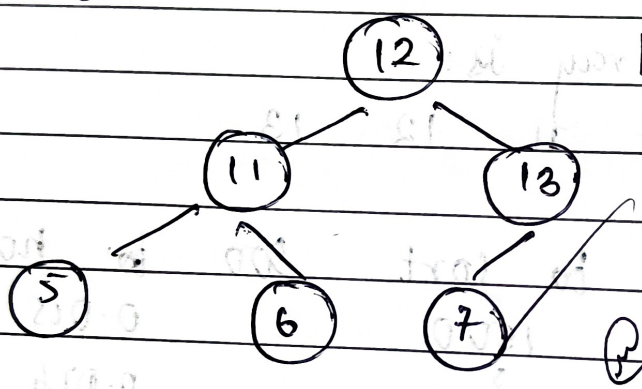
5 6 7 11 12 13

Time taken to sort 500 no. is 0.00

1500	0.015
2500	0.016
3500	0.016
4000	0.047
5500	0.078
6500	0.094
7000	0.124
8000	0.189
9000	0.234
10500	0.266
11500	0.312
12500	0.375
13000	0.407
14500	0.467



Tree:



[12, 11, 13, 5, 6, 7]