Heap Sort

```
#include <stdio.h>

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

void heapify (int arr[ ], int , n, int i ){
    int  largest  =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--)
    eh heapify (arr, n,i);

for (nt 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)
  printf ("%d", arr[i]);
    printf ("\n");
    }
int main (){
 int a [15000] n, i, j, ch, temp;
   clock_t start, end.
  while (1){
printf ("\n1: for manual Entry of N value
    and array elements");
printf ("\n2: for manual display time taken for
sorting number of element N in the range
   500 to 14500);
printf ("\n3; Enter Exit");
printf ("\n Enter your choice");
scanf ("%d", &n); switch (ch)
case 1; printf ("\n Enter your array
        Elements");
  for (i=0; i<n; i++) {
    scanf ("%d", &a[i]);
    }
start = clock ();
  heapsort (a,n);
  end = clock ();
  printf ("\n Sorted array ");
  for (i=0; i<n; i++){
    printf ("%d ", a[i]);
    printf ("\n Time taken to store %d
        number is %f secs", n,
((double) (end - start))/CLOCKS_PER_
    SEC));
    break;
case 2:
```

```
n= 500;
while (n<= 4500){
for (i=0; i<n; i++)
    a[i] = n-i
  }
start = clock ();
heapsort (a, n);
for (i=0; i < 500000; j++) {temp = 38/600 }}
    end = clock ();
printf ("\n Time taken to sort %d number
is %.f Secs", n, (((double)(end - start))/
clocks_PER_SEC));
    n= n+ 1000;
  }
break;
case 3:  exit(0);
}
getchar ();
}
}
```

Output: 1) To enter data into array manually

2) To display time taken for sorting number of n elements n in range 500 to 4500

3) To exit.

```
Enter your choice : 1
Enter the number of Elements : 3
Enter array Elements : 33
  Sorted array is : " 22 33
Time taken to sort 3 numbers is 0.000002 Sec
```

Enter your choice: 2

Time taken to sort 500 numbers is 0.000557 sec

Time taken to sort 1500 number is 0.000538 sec

Time taken to sort 2500 number is 0.000624 sec

Time taken to sort 3500 number is 0.000784 sec

Time taken to sort 4500 number is 0.000926 sec

Time taken to sort 5500 number is 0.001010 sec

Time taken to sort 6500 number is 0.001092 sec

Time taken to sort 7500 number is 0.001097 sec

Time taken to sort 8500 number is 0.001094 sec

Enter your choice: 3

2. Floyd's algorithm

```
#include <stdio.h>
#define INF 9999
void floydWarshall (int graph[][100], int v){
    int i, j, k;
    for (k=0; k<v; k++){
    for (i=0; i<v; i++){
        for (j=0; j<v; j++){
    if (graph[i][k] + graph[k][j] < graph[i][j])
    graph[i][j] = graph[i][k] + graph[k][j];
        }}
    }
Printf ("Shortest distances between every pair of
        vertices :\n");
    for (i= 0; i<v; i++){
        for (j=0; j<v; j++){
    graph if
    if (graph[i][j] == INF)
```

```
        Printf ("%4s", "INF");
    else
        Printf("%4d", graph[i][j]);
    }
    Printf("\n");
  }
}

int main() {
    int V, E;
    Printf("Enter the number of vertices:");
    Scanf("%d", &V);
Printf("Enter the number of edges:");
    Scanf("%d", &E);
    int graph[100][100];
for (int i=0; i<V; i++) {
for (int j=0; j<V; j++) {
    graph[i][j] = INF;
  }
}
Printf("Enter the edge and their weight:\n");
    for (int i=0; i<E; i++) {
        int u, v, w;
    Scanf("%d %d %d", &u, &v, &w);
        graph[u][v] = w;
    }
    for (int i=0; i<V; i++) {
        graph[i][i] = 0;
    }
    floydWarshall(graph, V);
    return 0;
}
```

Output: Enter the number of vertices : 4
Enter the number of edges: 6
Enter the edges of and their weights:

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

Shortest distances between every pair of vertices:

0	5	3	9
7	0	2	6
3	2	0	4
9	6	4	0

Graph → Hash sort-