20/6/24

LAB (5):

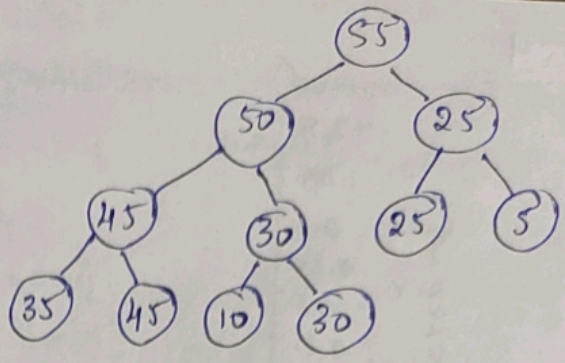
Heap sort:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void bottom_up_heapify (int n, int a[]){
    int p, item, c;
    for (p = (n-1)/2; p>=0; p--){
        item = a[p];
        c = 2*p+1;
        while (c<=n-1){
            if (c+1<=n-1 && a[c]<a[c+1]){
                c++;
            }
            if (item >= a[c]){
                break;
            }
            a[p] = a[c];
            p = c;
            c = 2*p+1;
        }
        a[p] = item;
    }
}
void swap (int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
void heap_sort (int n, int a[]){
    int i;
    bottom_up_heapify (n, a);
    for (i=n-1; i>=0; i--){
        swap (&a[0], &a[i]);
        bottom_up_heapify (i, a);
    }
}
```

```
int main()
{ int a[15000], n, i, j, ch, temp;
  clock_t start, end;
  while(1){
    printf("1. For manual entry of N value & array elements \n");
    printf("2. To display time taken for sorting no. of elements \n");
    printf("3. To exit");
    printf("Enter your choice: ");
    scanf("%d", &ch);
    switch(ch){
      case 1: printf("\n Enter the no. of elements: ");
              scanf("%d", &n);
              printf("Enter array elements: ");
              for(i=0; i<n; i++){
                  scanf("%d", &a[i]);
              }
              start = clock();
              heap_sort(n, a);
              end = clock();
              printf("Sorted array is: ");
              for(i=0; i<n; i++)
                  printf("%d", a[i]);
              printf("Time taken to sort %d no.s is %f secs", n
                  ((double)(end-start)) / CLOCKS_PER_SEC));
              break;

case 2: n = 500;
        while(n <= 14500){
            for(i=0; i<n; i++){
                a[i] = n-i;
            }
            start = clock();
            heap_sort(n, a);
            end = clock();
        }  n = n+1000;
```

```
        break;
    case 3: exit(0);
    }
    getchar();
    }
    return 0;
}
```

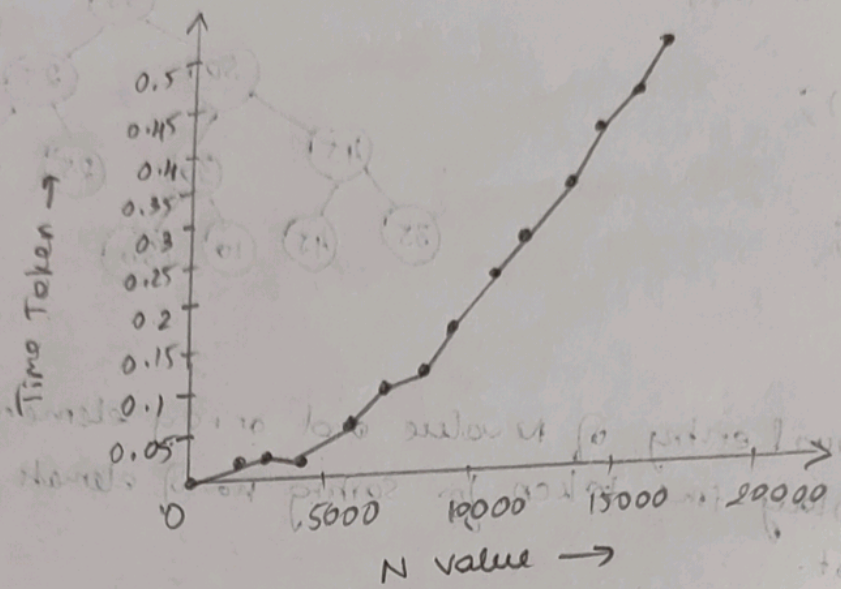


output:

1. For manual entry of N value and array elements
2. To display time taken for sorting no. of elements N
3. To exit.

Enter your choice: 1
Enter no. of elements: 11
Enter array elements: 5 35 25 45 30 55 25 45 50 10
Sorted array is: 5 10 25 25 30 30 35 45 45 50 55
Time taken to sort 11 no.s is 0.0 secs

Enter your choice: 2
Time taken to sort 500 no.s is 0.0 secs
" 1500 " 0.015 "
" 2500 " 0.016 "
" 3500 " 0.016 "
" 4500 " 0.047 "
" 5500 " 0.078 "
" 6500 " 0.094 "
" 7500 " 0.124 "
" 8500 " 0.188 "
" 9500 " 0.234 "
" 10500 " 0.266 "
" 11500 " 0.312 "
" 12500 " 0.375 "
" 13500 " 0.407 "
" 14500 " 0.468 "



Floyd's Algorithm :

```
#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 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 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
```