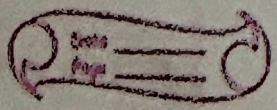
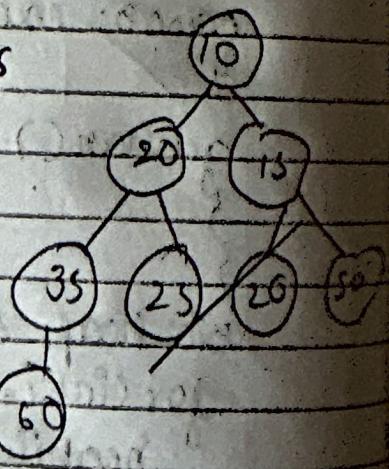
20/08/24 Heap sort

```
#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, i, j, ch;
  clock_t start, end;
  while(1){
    printf("\n1. For manual entry of N value & array");
    printf("\n2. To display time taken for sorting
          number of elements N");
    printf("\n3. To Exit");
    printf("\nEnter your choice");
    scanf("%d", &ch);
    switch(ch){
    case 1:
        printf("\nEnter the number of elements");
        scanf("%d", &n);
        printf("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 (((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(a,n);
     end=clock();
     printf("\n sorted array is ");
     print_array(a,n);
     printf("\n Time taken to sort %d, ((double)
     (end-start))/CLOCKS_PER_SEC));
     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 fprint_array (int a[], int n){
  for(int i=0; i<n; i++){
    fprintf ("%d ", a[i]);
  }
}
```



Output

1. For manual entry of N value & array elements
2. To display time taken for sorting numbers
3. To exit

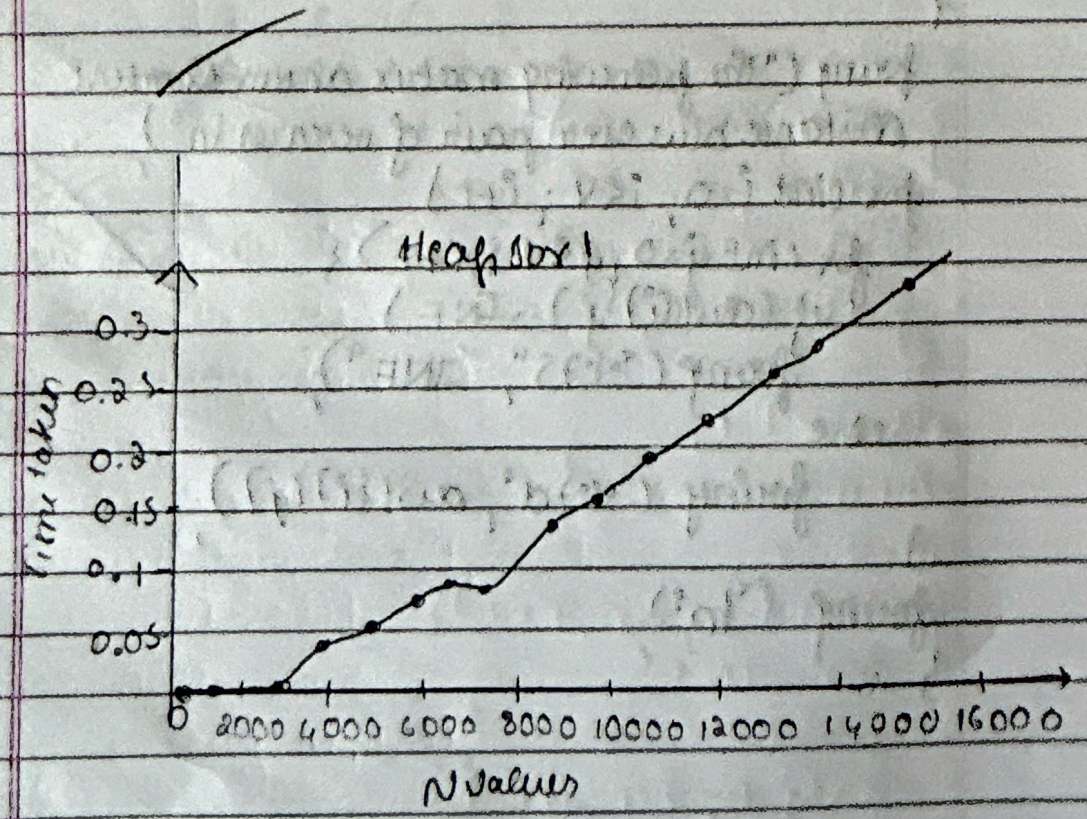
Enter your choice: 1

Enter the number of elements: 8

Enter array elements: 10 15 20 20 25 35 50 60

time taken to sort 8 nos is 0 secs

Enter your choice: 2
Time taken to sort 500 numbers is 0.016 secs
Time taken to sort 1500 numbers is 0.015 secs
Time taken to sort 2500 numbers is 0.033 secs
Time taken to sort 3500 numbers is 0.016 secs
Time taken to sort 4500 numbers is 0.031 secs
Time taken to sort 5500 numbers is 0.016 secs
Time taken to sort 6500 numbers is 0.016 secs
Time taken to sort 7500 numbers is 0.031 secs
Time taken to sort 8500 numbers is 0.015 secs
Time taken to sort 9500 numbers is 0.016 secs
Time taken to sort 10500 numbers is 0.032 secs
Time taken to sort 11500 number is 0.015 secs
Time taken to sort 12500 numbers is 0.031 secs
Time taken to sort 13500 numbers is 0.016 secs
Time taken to sort 14500 numbers is 0.15 secs
Time taken to sort 1500 numbers is 0 secs



Floyd Warshall Algorithm

```
#include<stdio.h>
#define V 4
#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
  distance 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)
        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

21/6/22