K. Sumanth
19113105
CSE 2B

# DATA STRUCTURES - ASSIGNMENT

1. Program for Bubble sort

```
#include <stdio.h>
# define MAX 100
int main ()
{
    int arr[MAX], limit ;
    int  i, j, temp ;
    Printf ("\t\t\t\t  BUBBLE SORT \n\n");
    Printf (" Enter total number of elements : ");
    scanf ( "%d", & limit);
    Printf(" Enter array elements :\n");
    for (i=0 ; i< limit ; i++)
    {
       printf (" Enter element %3d : ", i+1);
       scanf ("%d", & arr[i]);
    }
    for(i=0 ; i< (limit -1) ; i++) // sort elements in Ascending
                                                          order
    {
       for ( j=0; i< (limit - i -1) ; j++)
       {
          if (arr [j] > arr [j+1])
          {
             temp = arr[j];
             arr[j] = arr[j+1];
             arr[j+1]= temp;
          }
    }  }
    Printf(" The elements after bubble sorting in
              ascending order : \n");
    for(i=0 ; i< limit ; i++)
    {
        Printf("%d ", arr[i]);
    }
    Printf("\n");

    for(i=0 ; i< (limit -1) ; i++) //sort elements in
    {                                        Descending order
      for (j=0 ; j< (limit - i-1) ;j++)
      {
          if (arr[j] < arr[j+1])
          {
              temp = arr[j];
              arr[j] = arr[j+1];
              arr[j+1] = temp ;
          }
      }
    }
    Printf (" The elements after bubble sorting in
              descending order : \n");
    for (i=0 ; i< limit ; i++)
    {
        Printf("%d" , arr[i]);
    }
    Printf ("\n");
    return 0;
}
```

15, 95, 25, 75, 65, 5, 35, 85

1. 15 , ~~95~~ , ~~25~~ , ~~75~~ , ~~65~~ , ~~5~~ , ~~35~~ , ~~85~~

25 ~~95~~ ~~95~~ ~~95~~ ~~95~~ ~~95~~ $\boxed{95}$

75 65 5 35 85

2. 15 25 ~~75~~ ~~65~~ ~~5~~ ~~35~~ 85

65 ~~75~~ ~~75~~ 75 $\boxed{85}$

5 35

3. 15 25 ~~65~~ ~~5~~ ~~35~~ 75

5 ~~65~~ 65 $\boxed{75}$

35

4. 15 ~~25~~ ~~5~~ 35 $\boxed{65}$

5 25

5. ~~15~~ ~~5~~ 25 $\boxed{35}$

5 15

6. 5 15 $\boxed{25}$

7. 5 $\boxed{15}$

5 15 25 35 65 75 85 95

Average Time Complexity : $O(n^2)$

Worst Time Complexity : $O(n^2)$

Best Time Complexity : $O(n)$

2. Program for insertion sort :

```
#include <stdio.h>
#define MAX 100
int main ( )
{
    int data [MAX], n, temp, i, j ;
    Printf (" Enter total no. of elements : ");
    scanf (" %d ", &n);
    printf (" \n Enter elements : ");
    for (i=0 ; i<n ; i++)
    {
        scanf (" %d ", & data [i]);
    }
    for ( i=1 ; i<n ; i++)
    {
        temp = data[i] ;
        j = i-1 ;
        while ( temp < data[j] && j>=0)
        {
            data [j+1] = data [j];
            j = j-1 ;
        }
        data[j+1] = temp ;
    }
    Printf (" \n Sorted array : \n");
    for (i=0 ; i<n ; i++)
    {
        printf ("%d \t ", data[i]);
    }
    return 0 ;
```

10, 90, 20, 80, 30, 70, 30, 40

10 ~~90~~ ~~20~~ 80 30 70 30 40

10 20 ~~90~~ ~~80~~ 30 70 30 40

10 20 ~~80~~ ~~90~~ ~~30~~ 70 30 40

30 90

10 20 30 ~~80~~ ~~90~~ ~~70~~ 30 40

~~70~~ 90

10 20 30 ~~70~~ ~~80~~ ~~90~~ ~~30~~ 40

~~30~~ 90

~~30~~ 80

10 20 30 30 ~~70~~ ~~80~~ ~~90~~ ~~40~~

~~40~~ 90

~~40~~ 80

10 20 30 30 40 70 80 90

Average Time Complexity : $O(n^2)$

Worst Time Complexity : $O(n^2)$

Best Time Complexity : $O(n)$

3. Program for selection sort

```
#include <stdio.h>
#define MAX 100
{
    int array[MAX], n, pos, temp, i, j;
    Printf("\t\t\t SELECTION SORT \n\n ");
    Printf(" Enter total no. of elements : ");
    scanf("%d", &n);
    Printf("\n Enter the elements : \n ");
    for(i=0 ; i<n ; i++)
    {
        scanf("%d", & array[i]);
    }
    for (i=0 ; i < (n-1) ; i++)
    {
        pos = i ;
        for (j = i+1 ; j<n ; j++)
        {
            if (array [pos] > array [j])
                pos = j;
        }
        if (pos ! = i)
        {
            temp = array [i];
            array [i] = array [pos];
            array [pos] = temp;
        }
    }
    Printf("\n Sorted list in ascending order :\n");
    for ( i=0 ; i<n ; i++)
    {
        Printf ("%d \t" , array[i]);
    }
    return 0 ;
}
```

~~2~~, 4, 6, 7, 8, ~~1~~, 3, 5, 7, 10

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| i= 2 | 1 | ~~4~~ | 6 | 7 | 8 | ~~2~~ | 3 | 5 | 7 | 10 |
| i= 4 | 1 | 2 | ~~6~~ | 7 | 8 | 4 | ~~3~~ | 5 | 7 | 10 |
| i= 6 | 1 | 2 | 3 | ~~7~~ | 8 | ~~4~~ | 6 | 5 | 7 | 10 |
| i = 7 | 1 | 2 | 3 | 4 | ~~8~~ | 7 | 6 | ~~5~~ | 7 | 10 |
| i = 8 | 1 | 2 | 3 | 4 | 5 | ~~7~~ | ~~6~~ | 8 | 7 | 10 |
| i = 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 10 |
| i = 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ~~8~~ | ~~7~~ | 10 |
| i= 8 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 8 | 10 |

Average Time Complexity : $O(n^2)$

Worst Time Complexity : $O(n^2)$

Best Time Complexity : $O(n^2)$

4. Program for Quick sort

```
#include <stdio.h>
Void quick sort (int number[25], int first, int last)
{
    int i, j, pivot, temp;
    if( first < last)
    {
      Pivot = first ;
         i = first+1 ;
         j = last ;
       while (i<j)
       {
          while ( number[i] <= number [pivot] && i< last)
          i++ ;
          while (number [j] >= number [Pivot])
          j-- ;
          if ( i<j)
          {
              temp = number [i];
              number[i] = number [j];
              number[j] = temp;
          }
       }
       temp = number [Pivot];
       number [Pivot] = number[j];
        number[j] = temp;
       quick sort(number, first, j-1);
       quick sort (number, j+1 , last);
    }
}
```

```
void main
{
    int i, count, number[25];
    printf(" Enter total number of elements : ");
    scanf("%d", & count);
    printf("\n Enter %d elements : ", count);
    for(i=0; i<count; i++)
    {
      scanf("%d\n", & number[i]);
    }
    quick sort (number, 0, count-1);
    printf("\nThe sorted elements are \n");
    for(i=0 ; i< count ; i++);
    {
       printf(" %d", number[i]);
    }
}
```

Average Time Complexity : $O(n \log n)$

Worst Time Complexity : $O(n^2)$

Best Time Complexity : $O(n \log n)$

13, 25, 95, 34, 98, 23, 45, 65, 88, 9

Pivot

$\textcircled{13} \quad \underline{\overset{i}{25}} \quad 95 \quad 34 \quad 98 \quad 23 \quad 45 \quad 65 \quad 88 \quad \underline{\overset{j}{9}}$

$13 \quad \underline{\overset{j}{9}} \quad \underline{\overset{i}{95}} \quad 34 \quad 98 \quad 23 \quad 45 \quad 65 \quad 88 \quad 25$

→ 9 becomes pivot as 'j' crossed 'i'

$9 \quad \textcircled{13} \quad \underbrace{95 \quad 34 \quad 98 \quad 23 \quad 45 \quad 65 \quad 88 \quad 25}_{\text{Position-2}}$

Pivoted element

Position-1

Position-1 is in sorted order

<u>Position-2</u> :

Pivot

$\textcircled{95} \quad 34 \quad \underline{\overset{i}{98}} \quad 23 \quad 45 \quad 65 \quad 88 \quad \underline{\overset{j}{25}}$

$95 \quad 34 \quad 25 \quad 23 \quad 45 \quad 65 \quad \underline{\overset{j}{88}} \quad \underline{\overset{i}{98}}$ ⇒ 88 pivot

$88 \quad 34 \quad 25 \quad 23 \quad 45 \quad \underline{\overset{j}{65}} \quad \underline{\overset{i}{95}} \quad 98$ ⇒ 65 Pivot

65 ~~88~~ $\quad 34 \quad 25 \quad 23 \quad \underline{\overset{j}{45}} \quad \underline{\overset{i}{88}} \quad 95 \quad 98$ ⇒ 45 pivot

$45 \quad 34 \quad 25 \quad \underline{\overset{j}{23}} \quad \underline{\overset{i}{65}} \quad 88 \quad 95 \quad 98$ ⇒ 23 pivot

$\underline{\overset{j}{23}} \quad \underline{\overset{i}{34}} \quad 25 \quad 45 \quad 65 \quad 88 \quad 95 \quad 98$ ⇒ 34 pivot

$34 \quad 23 \quad \underline{\overset{j}{25}} \quad \underline{\overset{i}{45}} \quad 65 \quad 88 \quad 95 \quad 98$ ⇒ 25 pivot

$25 \quad \underline{\overset{j}{23}} \quad \underline{\overset{i}{34}} \quad 45 \quad 65 \quad 88 \quad 95 \quad 98$ ⇒ 23 pivot

$23 \quad 25 \quad 34 \quad 45 \quad 65 \quad 88 \quad 95 \quad 98$

Add position-1 & position-2

9 13 23 25 34 45 65 88 95 98

5. program for shell sort

```
#include <stdio.h>
# include <conio.h>
void main ()
{
    int arr[30], i, j,K, tmp, num;
    printf("\t \t \t  SHELL SORT \n\n");
    printf("Enter total no. of elements :");
    scanf ("%d", & num);
    for( K=0 ; k<num ; K++)
    {
        printf("\n Enter %d number : ", k+1);
        scanf(" %d" , & arr[k]);
    }
    for( i= num/2 ; i>0; i= i/2)
    {
        for(j= i ; j<num; j++)
        {
            for (k=j ; k>=0 ; k=k-i)
            {
                if (arr[k+i] > =arr [k])
                break;
                else
                {
                    tmp = arr[k];
                    arr [k] = arr[k+i];
                    arr[k+i]= tmp;
                }
            }
        }
    }
    for(K=0 ; K<num ; K++)
    {
        Printf("%d\t", arr[k]);
    }
    getch();
}
```

88, 45, 67, 23, 55, 22, 11, 66

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| n = 8 | ~~88~~ | ~~45~~ | ~~67~~ | 23 | ~~55~~ | ~~22~~ | ~~11~~ | 66 |
| $K = \frac{8}{2} = 4$ | ~~55~~ | 22 | ~~11~~ | 23 | ~~88~~ | 45 | ~~67~~ | 66 |
| $K = \frac{4}{2} = 2$ | 11 | 22 | ~~55~~ | ~~23~~ | ~~67~~ | ~~45~~ | ~~88~~ | ~~66~~ |
| $K = \frac{2}{2} = 1$ | 11 | 22 | 23 | ~~55~~ | ~~45~~ | ~~67~~ | ~~66~~ | 88 |
| | 11 | 22 | 23 | 45 | 55 | 66 | 67 | 88 |

Average Time Complexity : $O(n^2)$

Worst Time Complexity : $O(n^2)$

Best Time Complexity : $O(n \log n)$

## 6. Program for heap sort

```
#include<stdio.h>
int main()
{
    int heap[10], no, i, j, c, root, temp;
    Printf("\n \t \t \t HEAP SORT \n\n ");
    Printf(" Enter no. of elements :");
    Scanf(" %d ", & no);
    Printf(" Enter elements : \n ");
    for(i=0; i<no ; i++)
    {
        scanf(" %d " ,& heap[i]);
    }
    for(i=1 ; i<no ; i++)
    {
        c = i;
        do
        {
            root = (c-1)/2;
            if (heap[root]< heap[c])
            {
                temp = heap[root];
                heap[root] = heap[c];
                heap[c] = temp;
            }
            c = root;
        }
        while(c!=0)
    }
    printf("\n Heap array :");
    for(i=0 ; i<no; i++)
    {
        Printf("%d \t", heap[i]);
    }
    for(j=no-1 ; j>=0; j--)
    {
        temp = heap[0];
        heap[0]= heap[j];
        heap[j] = temp;
        root =0;
        do
        {
            c=2* root +1;
            if((heap[c] < heap[c+1]) && c< j-1)
                c++;
            if( heap [root] < heap[c] && c<j)
            {
                temp = heap[root];
                heap[root] = heap [c];
                heap [c] = temp;
            }
            root =c;
        }
        while (c >j);
    }
    printf("\n\n Sorted array is \n ");
    for(i=0 ; i<no ; i++)
    {
        printf(" %d\t", heap [i]);

    }
}
```

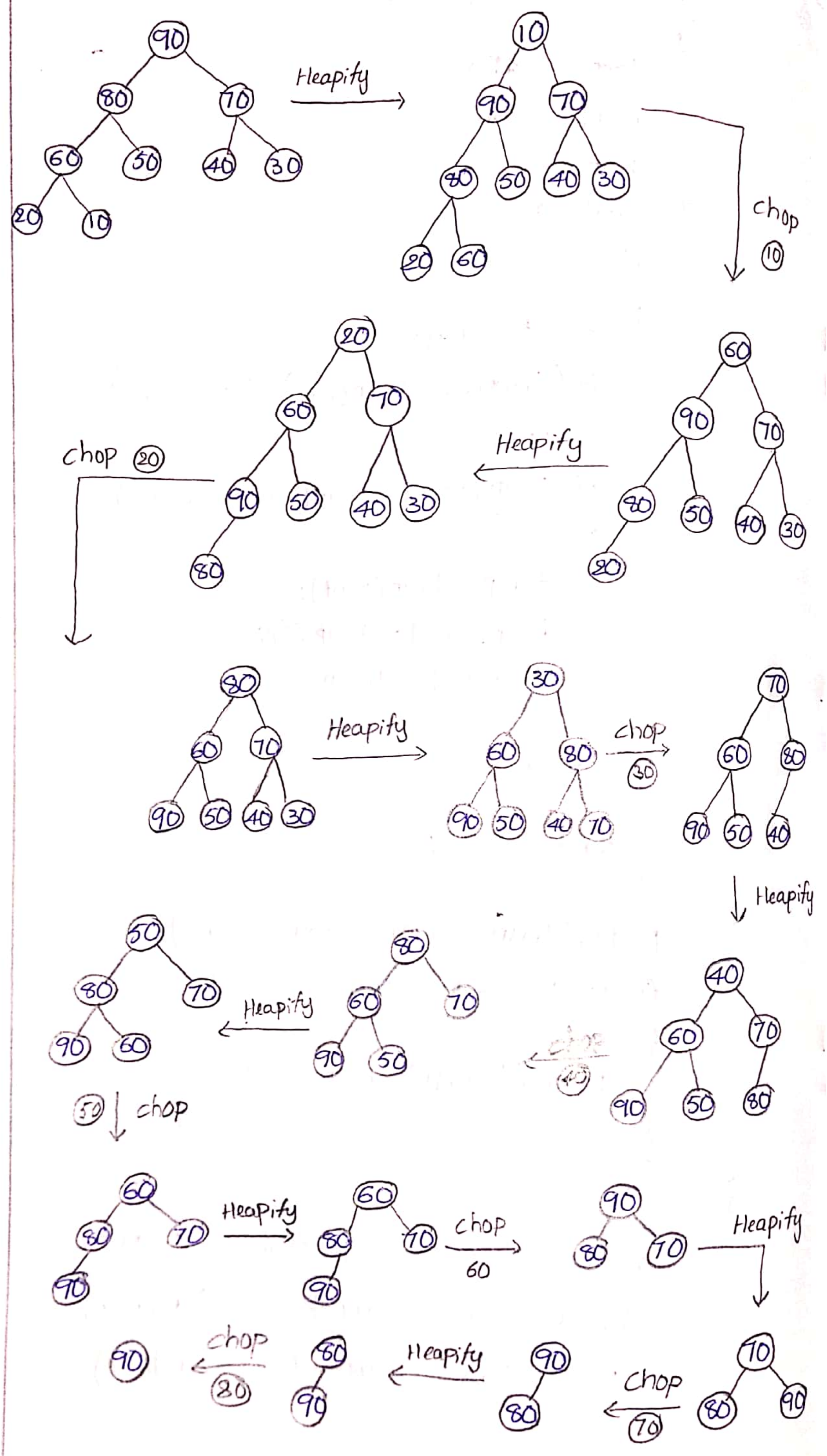
Average Time Complexity : O(n log n)

Worst Time Complexity : O(n log n)

Best Time Complexity : O(n log n)

90, 80, 70, 60, 50, 40, 30, 20, 10

10
20
30
40
50
60
70
80
90



7. Procedure for Merge sort :

Inputs :

input_file : Name of input file, input.txt

output-file : Name of output file, output.txt

run_size : Size of a run

num. ways : Number of runs to be merged

1. Read input_file such that at most 'run_size' elements are read at a time. Do following for the every run read in an array.

   a) Sort the run using Merge sort

   b) Store the sorted run in a temporary file, say 'i' for ith an.

2. Merge the sorted files

34, 45, 67, 88, 54, 12, 2, 19, 89, 43, 67, 100

a) When run size is 2

34 45 67 88 54 12 2 19 89 43 67 100

| 34 45 | 67 88 | 12 54 | 2 19 | 43 89 | 67 100 |

| 34 45 67 88 | 2 12 19 54 | 43 67 89 100 |

| 2 12 19 34 45 54 67 88 | 43 67 89 100 |

| 2 12 19 34 43 45 54 67 67 88 89 100 |

b) When run size is 3

34 45 67 88 54 12 2 19 89 43 67 100

34 45 67 88 54 12 2 19 89 43 67 100

---

b) When run size is 3

34 45 67 88 54 12 2 19 89 43 67 100

| 34 | 45 | 67 |   | 12 | 54 | 88 |   | 2 | 19 | 89 |   | 43 | 67 | 100 |

| 12 | 34 | 45 | 54 | 67 | 88 |   | 2 | 19 | 43 | 67 | 89 | 100 |

| 2 12 19 34 43 45 54 67 67 88 89 100 |

Average Time Complexity : O(n logn)

Worst Time Complexity : O (n logn)

Best Time Complexity : O (n logn)