ADVANCED ALGORITHM ANALYSIS

Final Term Exam

NAME :SALMAN AFRFIDI IDNO#14110

CORSE:MS(CS)

(Time Allowed: 06 hours)

Marks:50

Q1. Analyze Bubble Sort. (13 Marks)

ANSWER1: Bubble sort is one of the simplest sorting algorithms. The two adjacent elements of an array are checked and swapped if they are in wrong order and this process is repeated until we get a sorted array. The steps of performing a bubble sort are:

Compare the first and the second element of the array and swap them if they are in wrong order.

Compare the second and the third element of the array and swap them if they are in wrong order.

Proceed till the last element of the array in a similar fashion.

Repeat all of the above steps until the array is sorted.

This will be more clear by the following visualizations.

Initial array

16	19	11	15

First iteration

16	**19**	**11**	**15**
16	**19**	**11**	**15**

swap

16	**11**	**19**	**15**

16	**11**	**19**	**15**

swap

16	**11**	**15**	**19**

SECOUND ITERATION:

16	**11**	**15**	**19**

SWAP:

11	**16**	**15**	**19**

11	**16**	**15**	**19**

SWAP:

11	**15**	**16**	**19**

11	15	16	19

THIRD ITERATION:

11	15	16	19

11	15	16	19

11	15	16	19

No swap → Sorted → break the loop

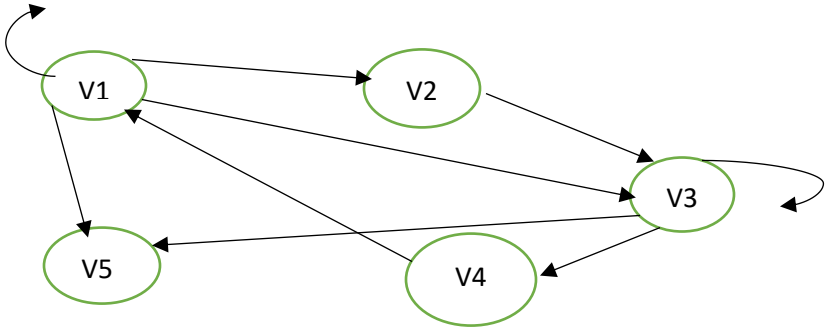
```
#include <stdio.h>
#include <math.h>

int main()
{
   int a[] = {16, 19, 11, 15, 10, 12, 14};
   int i,j;

   //repeating loop 7 (number of elements in the array) times
   for(j = 0; j<7; j++)
   {
      //initially swapped is false
      int swapped = 0;
      i = 0;
```

```
    while(i<7-1)
    {
        //comparing the adjacent elements
        if (a[i] > a[i+1])
        {
            //swapping
            int temp = a[i];
            a[i] = a[i+1];
            a[i+1] = temp;
            //Changing the value of swapped
            swapped = 1;
        }
        i++;
    }
    //if swapped is false then the array is sorted
    //we can stop the loop
    if (!swapped)
        break;
  }

  for(i=0;i<7;i++)
    printf("%d\n",a[i]);

  return 0;
}
```

In this code, we are just comparing the adjacents elements and swapping them if they are not in order. We are repeating this process 7 times (number of elements in the array). We have also assigned a variable 'swapped' and making it 1 (True) if any two elements get swapped in an iteration. If no interchanging of elements happens then the array is already

sorted and thus no change in the value of the 'swapped' happens and we can break the loop.

Q2. Design an Adjacency Matrix for the given graph. (12 Marks)



ANSWER 2: Definition of an Adjacency Matrix: An adjacency matrix is defined as follows: Let G be a graph with "n" vertices that are assumed to be ordered from v1 to vn.

The n x n matrix A, in which

aij= 1 if there exists a path from vi to vj

aij = 0 otherwise

is called an adjacency matrix.

Consider the following directed graph G (in which the vertices are ordered as v1, v2, v3, v4, and v5), and its equivalent adjacency matrix representation on the right:

	v1	v2	v3	v4	v5
v1	0	1	0	1	1
v2	0	0	0	1	0
v3	0	0	0	0	1
v4	0	0	0	0	0
v5	0	1	0	0	0

Graphs can also be represented in the form of matrices. The major advantage of matrix representation is that the calculation of paths and cycles can easily be performed using well known operations of matrices. However, the disadvantage is that this form of

representation takes away from the visual aspect of graphs. It would be difficult to illustrate in a matrix, properties that are easily illustrated graphically.

Using the matrix from the previous example and multiplying it by itself, we obtain the following new matrix:

Matrix representation of path of length 2

	v1	v2	v3	v4	v5
v1	**0**	**1**	**0**	**1**	**0**
v2	**0**	**0**	**0**	**0**	**0**
v3	**0**	**1**	**0**	**0**	**0**
v4	**0**	**0**	**0**	**0**	**0**
v5	**0**	**0**	**0**	**1**	**0**

The above matrix indicates that we can go from vertex v1 to vertex v2, or from vertex v1 to vertex v4 in two moves. In fact, if we examine the graph, we can see that this can be done by going through vertex v5 and through vertex v2 respectively. We can also reach vertex v2 from v3, and vertex v4 from v5, all in two moves.In general, to generate the matrix of path of length n, take the matrix of path of length n-1, and multiply it with the matrix of path of length 1.

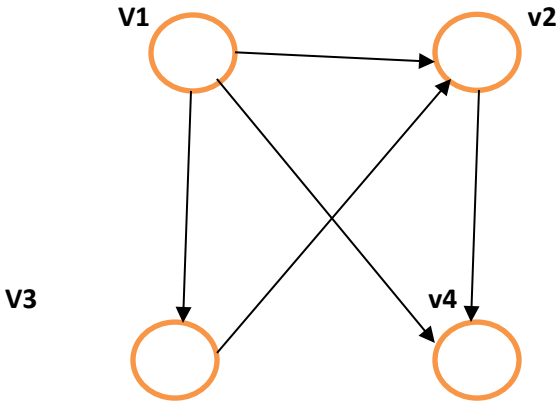
Q3. Consider the given Adjacency Matrix and design the graph. (12 Marks)

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

ANSWER: Consider the following directed graph G (in which the vertices are ordered as v1, v2, v3, and v4), and its equivalent adjacency matrix representation on the right:

	V1	V2	V3	V4
v1	**1**	**0**	**1**	**0**
v2	**0**	**1**	**1**	**1**
v3	**1**	**0**	**1**	**1**

v4 1 1 0 1



Q4. Design a Heap and then Sort using Heap Sort. (13 Marks)

10, 5, 31, 7, 11, 0

ANSWER: A heap is an array, visualized as a binary tree. Heaps tend to have the following methods: insert, pop/delete, and lookup/peek. Insert and pop have a runtime of O(log(n)), while peek is O(1). All levels of the binary tree must be filled from left to right. A heap is really great for quick access to the largest or smallest elements

10	**5**	**31**	**7**	**11**	**0**