Q.2 a. Design an algorithm for computing gcd (m,n) using Euclid's algorithm. Apply the algorithm to find gcd (31415, 14142).

Answer:

```
ALGORITHM Euclid(m, n) //Computes gcd(m, n) by Euclid's algorithm //Input: Two
    nonnegative, not-both-zero integers m and n //Output: Greatest common
    divisor of m and n while n=0
        do r ←m mod n
           m←n
           n←r
        return m
```

gcd(31415,14142) = gcd(14142,3131) = gcd(3131,1618) = gcd(1618,1513) = gcd(1513,105) = gcd(1513,105) = gcd(105,43) = gcd(43,19) = gcd(19,5) = gcd(5,4) = gcd(4,1) = gcd(1,0) = 1.

b. Let A be the adjacency matrix of an undirected graph. Explain what property of the matrix indicates that:
(i) the graph is complete.
(ii) the graph has a loop, i.e. an edge connecting a vertex to itself.
(iii) the graph has an isolated vertex, i.e. a vertex with no edges incident to it.

Answer:

For the adjacency matrix representation:
i. A graph is complete if and only if all the elements of its adjacency matrix except those on the main diagonal are equal to 1, i.e., $A[i,j]=1$ for every $1 \leq i,j \leq n$, $i = j$.
ii. A graph has a loop if and only if its adjacency matrix has an ele- ment equal to 1 on its main diagonal, i.e., $A[i,i]=1$for some $1 \leq i \leq n$.
iii. An (undirected, without loops) graph has an isolated vertex if and only if its adjacency matrix has an all-zero row.

Q.3 a. Write general outlines for analysing time efficiency of recursive algorithms.

Answer:

General Plan for Analyzing the Time Efficiency of Non-recursive Algorithms
1. Decide on a parameter (or parameters) indicating an input's size.
2. Identify the algorithm's basic operation. (As a rule, it is located in the inner-most loop.)

3. Check whether the number of times the basic operation is executed depends only on the size of an input. If it also depends on some additional property, the worst-case, average-case, and, if necessary, best-case efficiencies have to be investigated separately.
4. Set up a sum expressing the number of times the algorithm's basic operation is executed.
5. Using standard formulas and rules of sum manipulation, either find a closed-form formula for the count or, at the very least, establish its order of growth.

b. Design a recursive algorithm for computing 2^n for any non negative integer n that is based on the formula: $2^n = 2^{n-1} + 2^{n-1}$. Draw a tree of recursive calls for this algorithm for 2^4.

Answer:

```
Algorithm Power(n)
    //Computes $2^n$ recursively by the formula $2^n = 2^{n-1} + 2^{n-1}$
    //Input: A nonnegative integer n
    //Output: Returns $2^n$
    if n =0 return 1
    else return Power(n−1) + Power(n−1)
```

Q.4 a. Write the algorithm for Bubble sort and derive its time complexity.

Answer:

```
void bubbleSort(int* array,int size)
{
int swapped;
int i;
for(i =1; i < size; i++)
{
      swapped =0;//this flag is to check if the array is already sorted
int j;
for(j =0; j < size - i; j++)
{
if(array[j]> array[j+1])
{
int temp = array[j];
            array[j]= array[j+1];
            array[j+1]= temp;
            swapped =1;
}
}
if(!swapped){
break;//if it is sorted then stop
}
}
}
```

Complexity of bubble sort

• For an array of size n, in the worst case:

1st passage through the inner loop: n-1 comparisons and n-1 swaps

• (n-1)st passage through the inner loop: one comparison and one swap.

• All together: c ((n-1) + (n-2) + ... + 1), where c is the time required to do one comparison, one swap, check the inner loop condition and increment j.

• We also spend constant time k declaring i,j,temp and initialising i. Outer loop is executed n-1 times, suppose the cost of checking the loop condition and decrementing i is c1

c ((n-1) + (n-2) + ... + 1) + k + c1(n-1)

(n-1) + (n-2) + ... + 1 = n(n-1)/2

so our function equals

c n*(n-1)/2 + k + c1(n-1) = 1/2c (n2-n) + c(n-1) + k

complexity O(n2).

b. Explain how Binary Search method fails to find 43 in the given sorted array:
8, 12, 25, 26, 35, 48, 57, 78, 86, 93, 97, 108, 135, 168, 201 (8)

Answer:

The value to be searched is 43. As explained earlier, in the first iteration

low = 1, high = 15 and mid = 8

As 43 < A[8] = 78, therefore, as in part (i)

low = 1, high = 8 ─ 1 = 7 and mid = 4

As 43 > A[4] = 26, the algorithm makes another iteration in which

low = mid + 1 = 5 high = 7 and (new) mid = (5 + 7)/2 = 6

Next, as 43 < A[6] = 48, the algorithm makes another iteration, in which

low = 5 high = 6 ─ 1 = 5 hence mid = 5, and A[5] = 35 As 43 > A[5], hence value ≠ A[5].

But, at this stage, low is not less than high and hence the algorithm returns ─ 1, indicating failure to find the given value in the array.

Q.5 a. Apply the DFS-based algorithm to solve the topological sorting problem for the following diagram: (8)

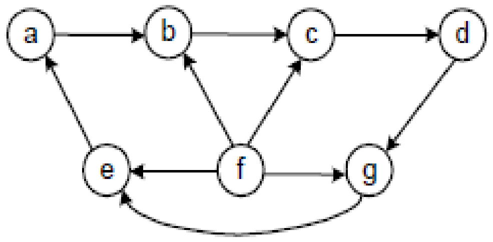
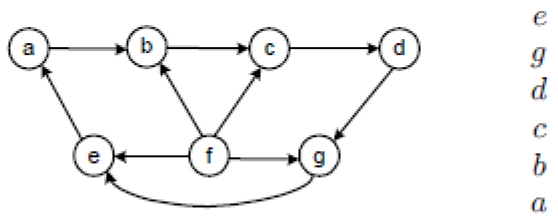


Fig.2

Answer:

The digraph below is not a dag. Its DFS traversal that starts at aencounters a back edge from e to a:



b. **Explain *Lexicographic Permute* algorithm, which generates permutations in lexicographical order. Also generate all permutations of {1, 2, 3, 4} by this algorithm. (8)**

Answer:

```
ALGORITHM LexicographicPermute(n)
//Generates permutations in lexicographic order
//Input: A positive integer n
//Output: A list of all permutations of {1, . . . , n} in lexicographic order
initialize the first permutation with 12 . . . n
while last permutation has two consecutive elements in increasing order do
let i be its largest index such that ai < ai+1 //ai+1> ai+2 > . . . > an
find the largest index j such that ai < aj //j ≥ i + 1 since ai < ai+1
swap ai with aj //ai+1ai+2 . . . an will remain in decreasing order
reverse the order of the elements from ai+1 to an inclusiveadd the new permutation to the list
```

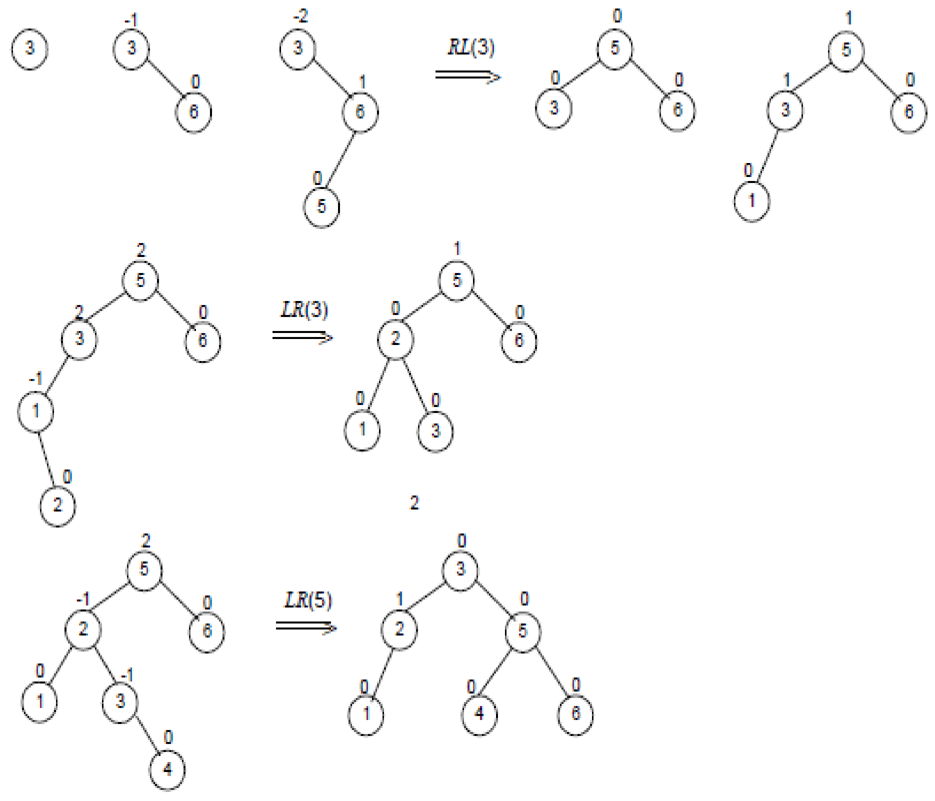
The permutations of {1, 2, 3, 4} generated in lexicographic order. (Readhorizontally.)

1234 1243 1324 1342 1423 1432
2134 2143 2314 2341 2413 2431
3124 3142 3214 3241 3412 3421
4123 4132 4213 4231 4312 4321

Q.6 a. Define AVL tree. Construct an AVL tree for the list 3, 6, 5, 1, 2, 4.

Answer:

An *AVL tree* is a binary search tree in which the *balance factor* ofevery node, which is defined as the difference between the heights of the node'sleft and right subtrees, is either 0 or +1 or −1.



b. Explain the idea behind using Gaussian Elimination method to solve a system of linear equation. Solve the following system by Gaussian elimination.

$$x1 + x2 + x3 = 2$$
$$2x1 + x2 + x3 = 3$$
$$x1 - x2 + 3x3 = 8$$

(8)

Answer:

$$\begin{bmatrix} 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 3 \\ 1 & -1 & 3 & 8 \end{bmatrix} \quad \begin{array}{l} \text{row 2 - } \frac{2}{1}\text{row 1} \\ \text{row 3 - } \frac{1}{1}\text{row 1} \end{array}$$

$$\begin{bmatrix} 1 & 1 & 1 & 2 \\ 0 & -1 & -1 & -1 \\ 0 & -2 & 2 & 6 \end{bmatrix} \quad \text{row 3 - } \frac{-2}{-1}\text{row 2}$$

$$\begin{bmatrix} 1 & 1 & 1 & 2 \\ 0 & -1 & -1 & -1 \\ 0 & 0 & 4 & 8 \end{bmatrix}$$

Then, by backward substitutions, we obtain the solution as follows:

$x_3 = 8/4 = 2,\ x_2 = (-1 + x_3)/(-1) = -1,$ and $x_1 = (2 - x_3 - x_2)/1 = 1.$

Q.7 a. Apply Warshall's algorithm to find the transitive closure of the digraph defined by the following adjacency matrix:

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

Answer:

$$R^{(0)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad R^{(1)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(2)} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad R^{(3)} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(4)} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = T$$

b. Differentiate between Spanning tree and minimum spanning tree (MST). Apply Prim's algorithm to find MST for the following graph.

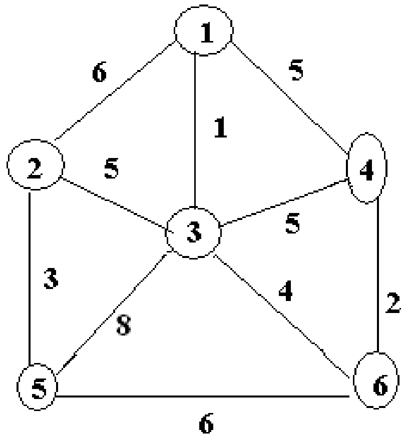
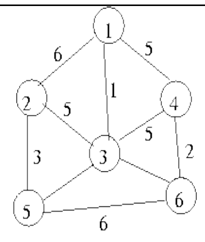
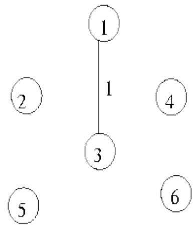


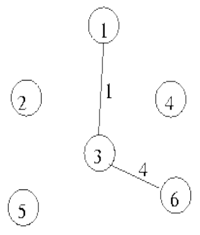
Fig.3

Answer:

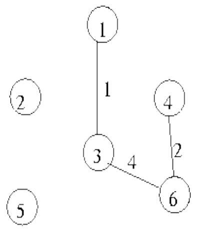




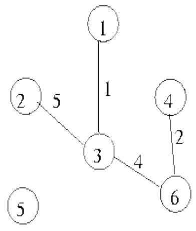
Iteration 1. U = {1}



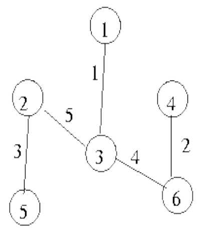
Iteration 2. U = {1,3}



Iteration 3. U = {1,3,6}



Iteration 4. U = {1,3,6,4}



Iteration 5. U = {1,3,6,4,2}

Q.8 a. Write algorithm for Counting Sort. Calculate the time efficiency of this algorithm. (8)

Answer:

ComparisonCountingSort(A[0..n−1]) //Sorts an array by comparison counting //Input: An array A[0..n−1] of orderable elements //Output: Array S[0..n−1] of A's elements sorted in nondecreasing order for i ←0 to n−1 do Count[i]←0 for i ←0 to n−2 do for j ←i +1 to n−1 do if A[i]<A [j] Count[j]←Count[j]+1 else Count[i]←Count[i]+1 for i ←0 to n−1 do S[Count[i]]←A[i] return S

What is the time efficiency of this algorithm? It should be quadratic because the algorithm considers all the different pairs of an n-elementarray. More formally, the number of times its basic operation, the comparison A[i]<A [j], is executed is equal to the sum we have encountered several times already:

b. Write notes on the following: (4×2)

(i) P and NP problems
(ii) CNF-satisfiability problem

Q.9 a. Write Pseudocode for the Bisection method. Explain the strength and weaknesses of this method. (4+2+2)

Answer:

```
Bisection(f (x), a, b, eps, N)
//Implements the bisection method for finding a root of f (x) = 0
//Input: Two real numbers a and b, a < b,
// a continuous function f (x) on [a, b], f (a)f (b) <0,
// an upper bound on the absolute error eps >0,
// an upper bound on the number of iterations N
//Output: An approximate (or exact) value x of a root in (a, b)
//or an interval bracketing the root if the iteration number limit is reached
        n←1 //iteration count
        while n ≤ N do
        x ←(a + b)/2
        if x −a <eps return x
        fval←f (x)
        if fval = 0 return x
        if fval ∗f (a)<0
        b←x
        else a ←x
        n←n + 1
        return “iteration limit”, a, b
```

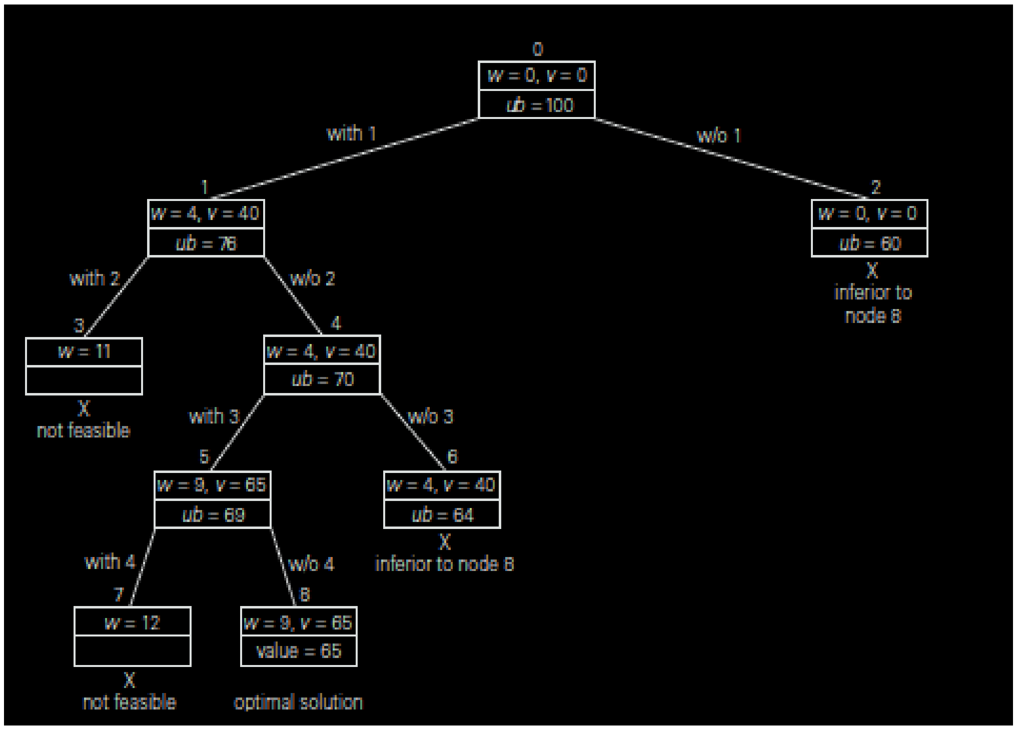
It always converges to a root whenever we start with an interval whose properties are very easy to check. And it does not use derivatives of the function *f (x)* as some faster methods do. The principal weakness of the bisection method as a general algorithm for solving equations is its slow rate of convergence compared with other known methods. It is for this reason that the method is rarely used. Also, it cannot be extended to solving more general equations and systems of equations.

b. Solve the following instance of the Knapsack problem by the branch and bound algorithm: (8)

Item	**Weight**	**Value**	**Value / Weight**
1	**4**	**$40**	**10**
2	**7**	**$42**	**6**
3	**5**	**$25**	**5**
4	**3**	**$12**	**4**

The Knapsack's capacity w is 10.

Answer:



At the root of the state-space tree, no items have been selected as yet. Hence, both the total weight of the items already selected *w* and their total value *v* are equal to 0. The value of the upper bound computed is \$100. Node 1, the left child of the root, represents the subsets that include item 1. The total weight and value of the items already included are 4 and \$40, respectively; the value of the upper bound is 40 + *(*10 − 4*)* ∗6 = \$76.
Node 2 represents the subsets that do not include item 1. Accordingly, *w* = 0, *v* = \$0, and *ub* = 0 + *(*10 − 0*)* ∗6 = \$60. Since node 1 has a larger upper bound thanthe upper bound of node 2, it is more promising for this maximization problem,and we branch from node 1 first. Its children—nodes 3 and 4—represent subsetswith item 1 and with and without item 2, respectively. Since the total weight *w* ofevery subset represented by node 3 exceeds the knapsack's capacity, node 3 canbe terminated immediately. Node 4 has the same values of *w* and *v* as its parent;the upper bound *ub* is equal to 40 + *(*10 − 4*)* ∗5 = \$70. Selecting node 4 over node2 for the next branching, we get nodes 5 and 6 by respectively includingand excluding item 3. The total weights and values as well as the upper bounds forthese nodes are computed in the same way as for the preceding nodes. Branchingfrom node 5 yields node 7, which represents no feasible solutions, and node 8, which represents just a single subset {1, 3} of value \$65. The remaining live nodes 2 and 6 have smaller upper-bound values than the value of the solution represented by node 8. Hence, both can be terminated making the subset {1, 3} of node 8 the optimal solution to the problem.

TEXT BOOK

I. Introduction to The Design & Analysis of Algorithms, Anany Levitin, Second Edition, Pearson Education, 2007