Ashesi University
Mid-term examination - 29h February, 2024
Time : 90 minutes
TOTAL marks = 80

INSTRUCTIONS: Answer all questions. Time allowed. 90 m

Question 1.ANALYSIS

Consider the following algorithm. Using the summation method, find the time complexity of the function MODROOTNEW (We shall assume that **function** = **Algorithm** in this case) and that function MODNEW() takes constant time.

```
function MODNEW(a, N)
    r ← ⌊a/N⌋
    return a − (r×N)
end function

function MODROOTNEW(y, N)
    a ← 0
    b ← N
    while a ≤ b do
        q ← ⌊ N/2 ⌋
        if MODNEW(q², N) = y then
            return q
        else if MODNEW(q², N) < y then
            a ← q + 1
        else
            b ← q − 1
        end if
    end while
end function
```

[8]

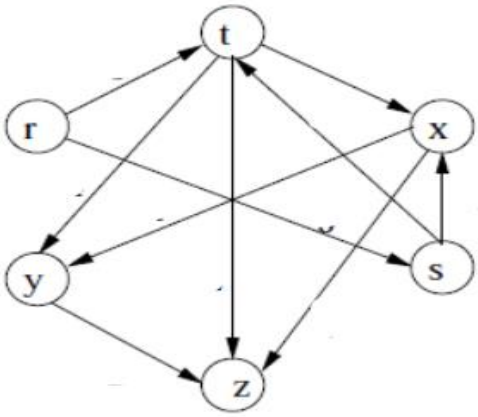
$$\sum_{i=1}^{\log n} 1 = O(\log_2 n)$$

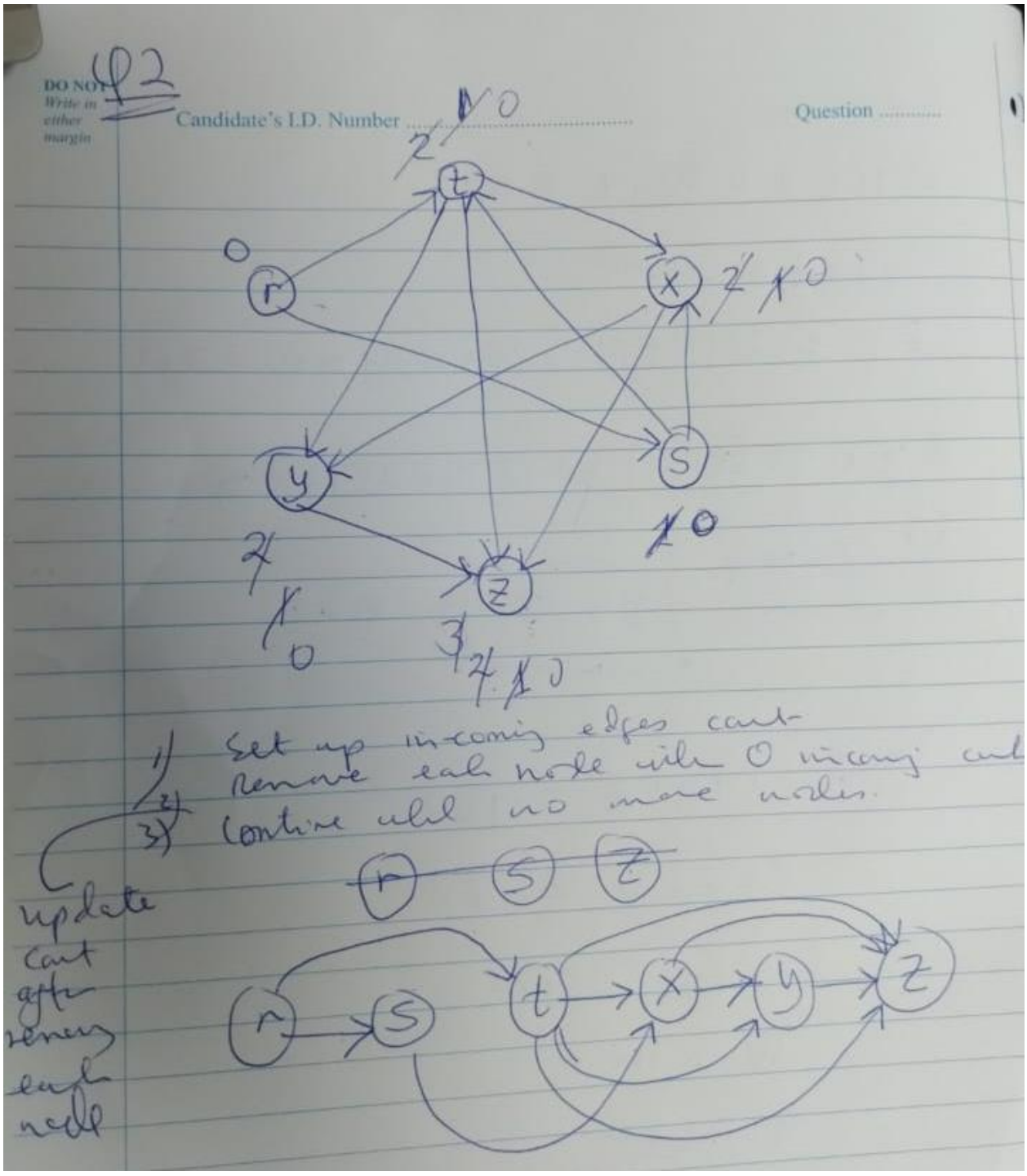
This questions was ambiguous . It could also give you O(n) but actually it is not an algorithm since it does not terminate in the worst case. So we factored this into the % that was awarded to everyone

Question 2. TOPOLOGICAL SORT

For the graph shown below, provide a topological sort algorithm using the source removal method, and provide the time complexity as well as the class asymptotic growth rate this time complexity belongs to. Show all workings at every step. Do not jump or skip any step. How many topological sorts can be obtained from this graph? List them all, but provide only one detailed work for ONLY one such sort.

SOLUTION ; ONLY ONE TOPOLOGICAL SORT - 2 MARK
LISTING r -s- t- x-y-z – 3 MARKS
DETAIL WORK - 5 MARKS


t
r
x
y
s
z

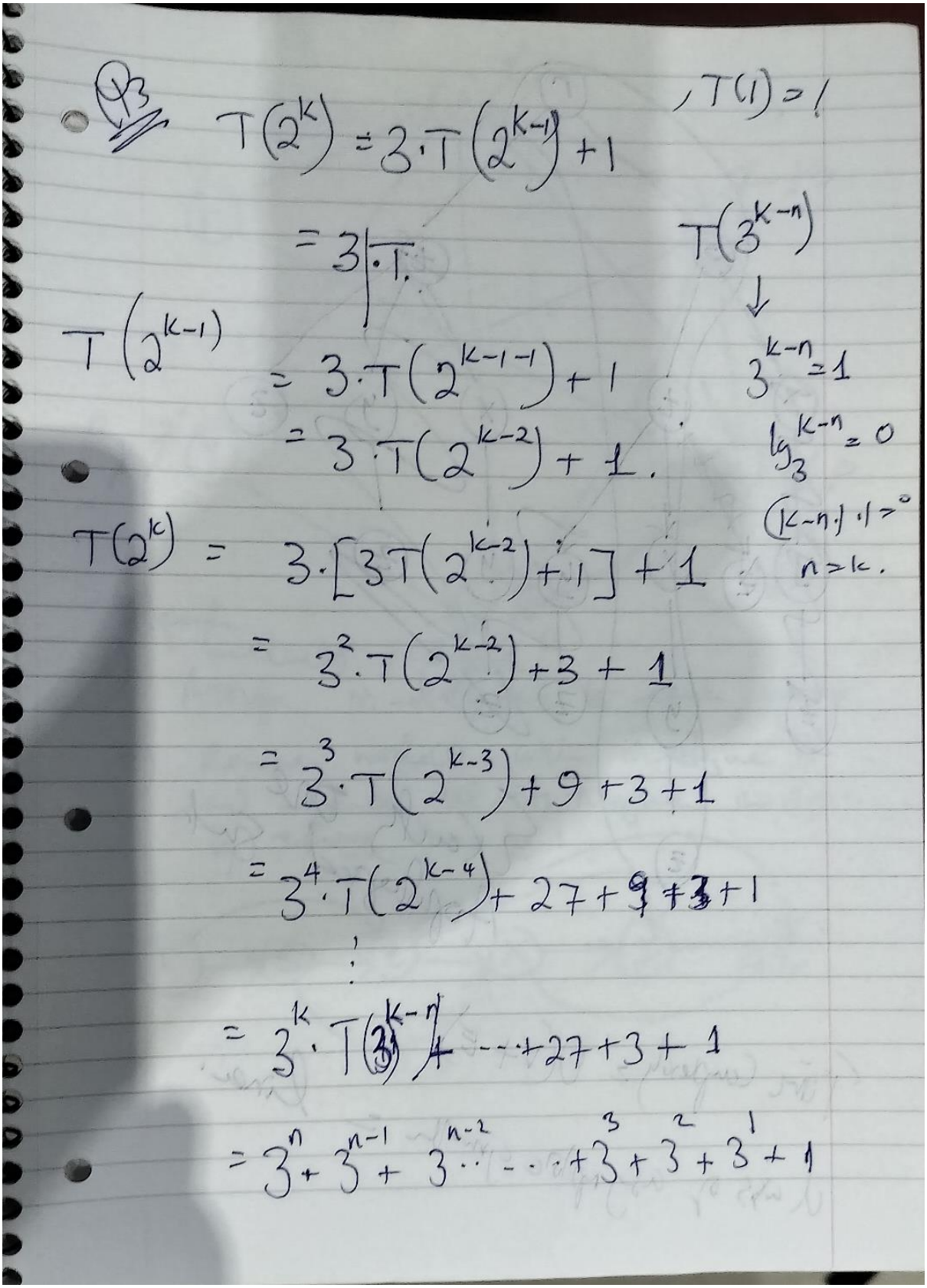

DO NOT Write in either margin
Candidate's I.D. Number
Question
Set up incoming edges cant
Remove each node with 0 incoming edg
Continue until no more nodes.
update cant after removing each node

[10]

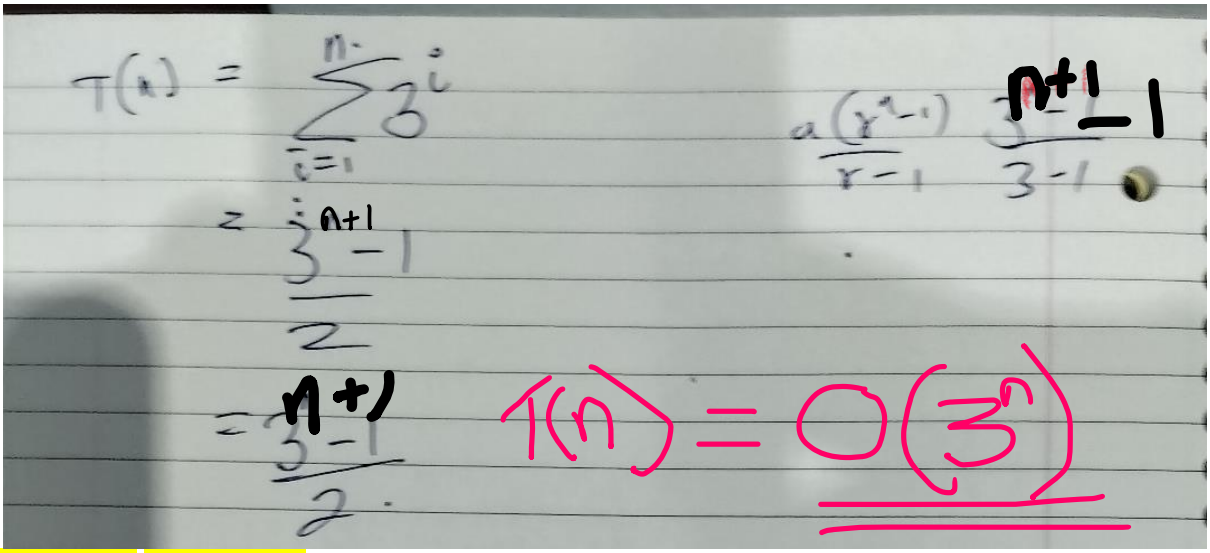
Question 3 RECURRENCE RELATION

Solve the following recurrence equation $T(2^k) = 3\,T(2^{k-1}) + 1$, $T(1) = 1$, using the substitution method as discussed in class.

[10]

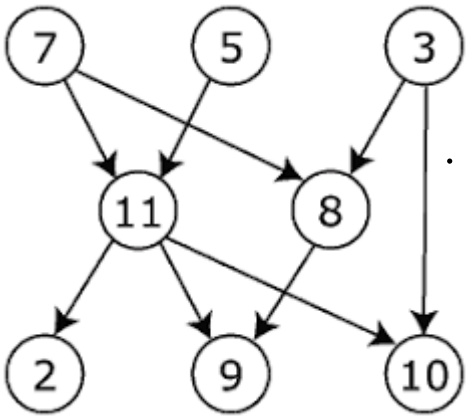


Ashesi University
Mid-term examination - 29h February, 2024
Time : 90 minutes
TOTAL marks = 80



Question 4 DFS & BFS

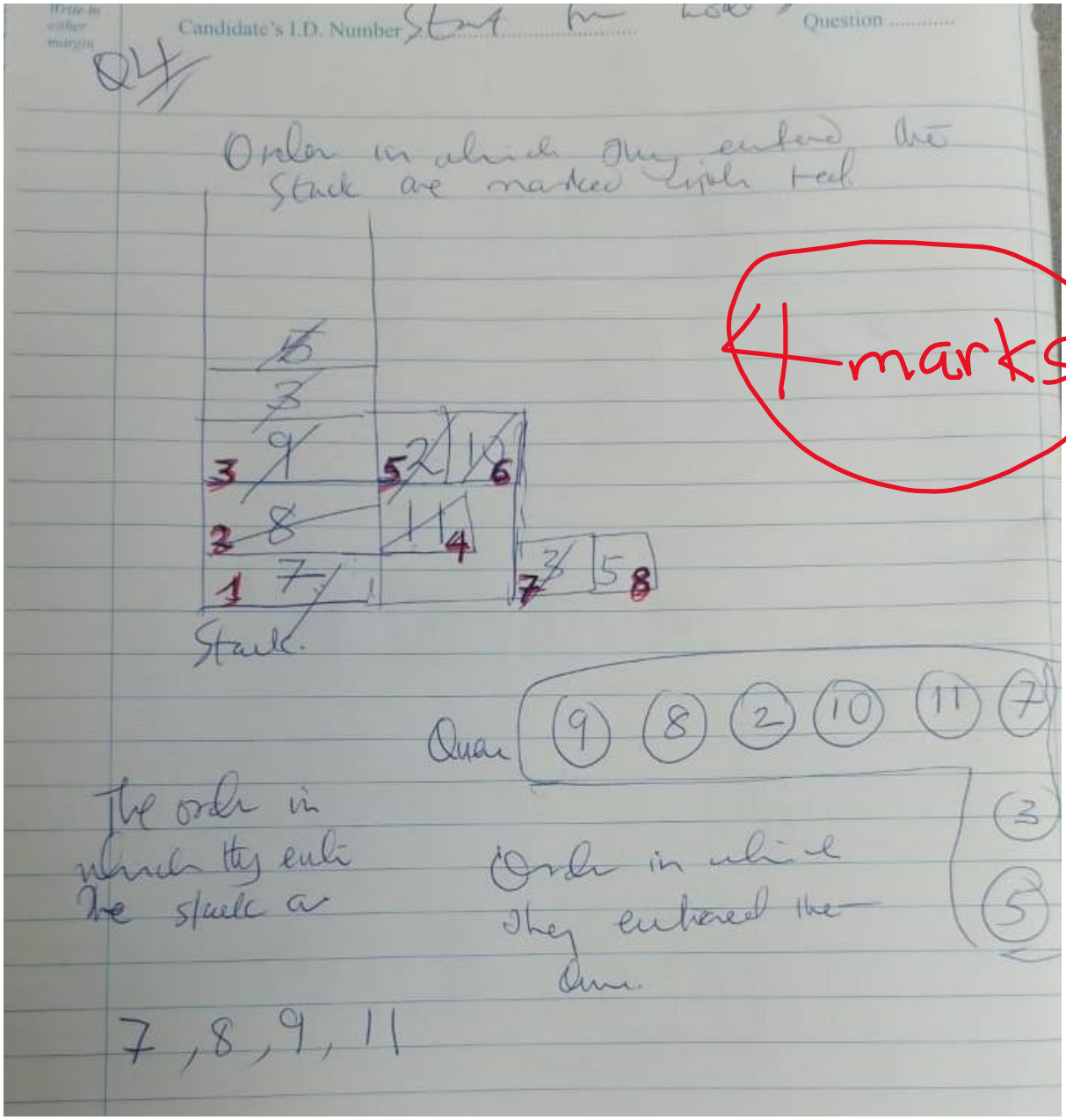
Consider the following graph.



a. In what order will the nodes be visited using a Breadth First Search starting from node 7 ? Show working, if necessary, with drawings

7, 8, 11, 9, 2, 10, 3, 5 —— 4 marks

b. In what order will the nodes be visited using a Depth First Search? Show working, if necessary, with drawings. We want to see the order in which the nodes enter the stack and the order in which they leave the stack and enter the queue.



[4,4 =8]

Question 5 DECREASE-AND-CONQUER

5.1 Which of these are/is not types of decrease and conquer

A. decrease by a constant
B. decrease by one
C. decrease by a constant factor
D. variable size decrease

5.2 binary search is an example of...

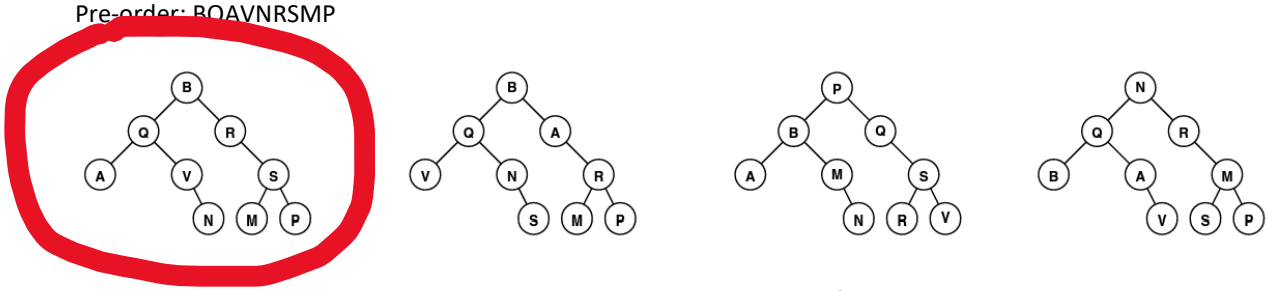
A. decrease by one example
B. decrease by a constant example
C. decrease by a constant factor example
D. None pf the above

5.3 Topological sort is (select which is wrong)

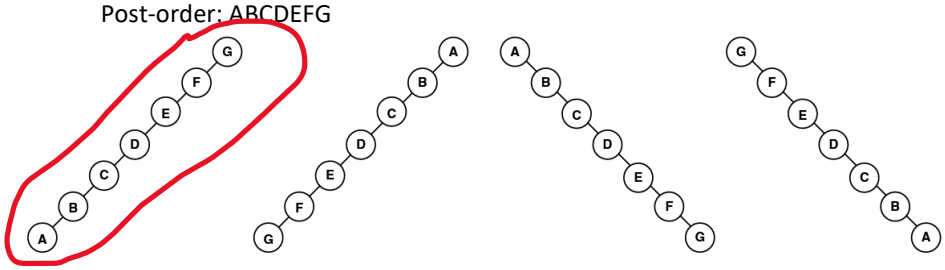
A. example of decrease by a constant
B. variable size decrease

C. working only if graph is a DAG
D. None of the above

5.4 Circle the correct binary tree (not necessarily a BST) that would produce both of the following traversals:
In-order: AQVNBRMSP
Pre-order: BQAVNRSMP



5.5 Circle the correct Binary Search Tree that would produce the following traversal:
Post-order: ABCDEFG



Some people also argued that D is also a legitimate answer. This was also factored into the 13 percent.

[1 each = 5]

Question 6 DIVDE & CONQUER

```
void Triangulate(VertexList vlist, TriangleList tlist)
{
    n = vlist.size;
    if (n == 3) {
        tlist.Add(vlist(0), vlist(1), vlist(2));
        return;
    }

    for (i0 = 0; i0 < n; i0++) {
        for (i1 = 0; i1 < n; i1++) {
            if (IsDiagonal(vlist, i0, i1))  {
                Split(vlist,sublist0, sublist1);
                Triangulate(sublist0, tlist);
                Triangulate(sublist1, tlist);
                return;
            }
        }
    }
}
```

Consider the above algorithm for performing a certain task, which is a divide and conquer algorithm. Assume that `IsDiagonal()` method/function takes O(n) time complexity, `Split()` is assume to just divide the `vlist` into 2 sub lists, and it is assumed to take a constant time (in real world this may not be so) .

a. Write down the base case for this recursive algorithm.

T(n) = 1, for n <=3

b. Write down the recursive part of the recurrence relation.

$2T(n/2) + \Theta(n^3)$

Reason for $\Theta(n^3)$ is that the IsDiagonal() method takes O(n) plus the two previous for loop, making $\Theta(n^3)$

c. Deduce the final recurrence relation for the above divide and conquer algorithm.

$$T(n) = \begin{cases} 1 & n <= 3 \\ 2T(n/2) + \Theta(n^3) & \end{cases}$$

d. Solve the recurrence relation you obtained in b. above.

By the Master;s theorem, we have a = 2, b = 2 , k = 3, and p = 0;
$a=2 < b^k = 2^3$ hence case 3(a)

Therefore $T(n) = \Theta(n^k \log^p n) = \Theta(n^3 \log^0 n) = \Theta(n^3)$

e. What is the time complexity for the above algorithm based on your answer to d.

Time |Complexity based on my answer to d is T(n) = $\Theta(n^3)$

[a=1, b=1,c=3,d=4,e=1 Total =10]

Question 7 BRUTE FORCE

Let A = [a_1, a_2, ... a_n] be an array of integers. We would like to find *d*, the distance between the two closest numbers in the array. The distance between any two numbers a_i and a_j is $|a_i - a_j|$. For example, if A = [9, 14, 25, 6, 16], then *d* = 2.

Remember a brute force approach is a straightforward way of solving a problem (what the teacher calls the "poor man's way " of solving a problem)

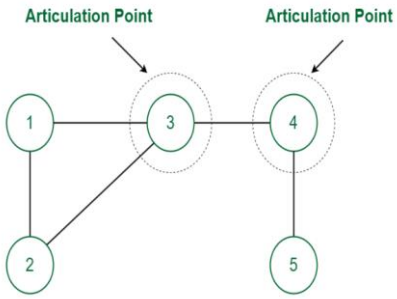
Design a **brute force** algorithm to compute *d*. How many times does this algorithm compare two distances?. Here design means provide an algorithm in the form of a pseudocode. Remember to write down your time complexity.

```
Algorithm findClosestNumbers(int[] A )
    d ← maxInt; // some huge no
    for (in j ←  0; j < n-2; j++) do
          for ( int k ← l; k <n; k++  )  do
                if  ((A[j] -  A[k] ) < d)
                  d ←  A[j] – A[k];
                <fi>
          <od>
    <od>
```

[15]

Question 8. MISCELLANEOUS

We define an articulation point as a vertex that when removed causes a connected graph to become disconnected. For this problem, we will try to find the articulation points in an undirected graph G. An example of an articulation points are shown in the graph below



a. How can we efficiently check whether or not a graph is disconnected? Suggest the name or names of an algorithm that can do this.

Time complexity = O(V+E)

b. Briefly describe an algorithm that uses a brute force approach to find all the articulation points in G in O(V(V+E)) time.

```
Algorithm FindArticulationPoints (V)    ----O(V)
  for <each vertex u in G do >
    <remove u from G and all its edges >
    <count how many components in the graph> --O(V+E)
     <mark u and put it back into the graph>
   <od>
Total Complexity :: O(V(V+E))
```

[2, 6, = 8]

Question 9. Write a recursive algorithm that returns the height of a binary tree. Argue that worst case running time of your algorithm is O(n)

[6]

```
Algorithm findHeight(Node root)
  if (root == null)
       return -1  //non-existence tree
  else
    return max(findHeighit(root.left), findHeight(root.right)) +1
```

OR

```
Algorithm  height(Node root) {
        if (root == null)
              return -1;
        int leftHeight = height(root.left);
        int rightHeight = height(root.right);
        return Math.max(leftHeight, rightHeight) + 1;
    }
```