INSTRUCTIONS: Answer all questions.

**Question 1.**

Find the GCD of x, y and z (where x = 203, y = 91 and z = 77) using Euclid’s Algorithm. A gcd code has been provided below in case you might need it.

int gcd(int u, int v)

{

if (v == 0)

return u;

else

return gcd(v, u % v);

}

[8]

Note that gcd(a,b,c) = gcd(a,gcd(b,c|)) = gcd(b,gcd(a,c)) = gcd(c,gcd(a,b))

So we select any one and do it.

Let us select gcd(a, gcd(b,c)), taking a = 203, b = 91, and c = 77.

Hence gcd(203, 91, 77) = gcd(203, gcd(91,77))

gcd(91,77) = gcd(77, 91 %77)

= gcd(77,14)

= gcd(14, 77%14)

= gcd(14,7)

= gcd (7, 14%7) = gcd(7,0) = 7

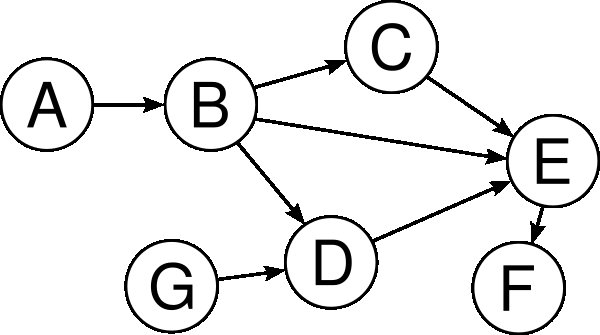
Therefore we have gcd(203,91,77) = gcd(203,7)

gcd(203,7)= gcd(7,203%7) = gcd(7,0) = 7

Therefore gcd(203,91,77) = 7

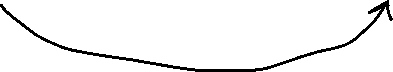
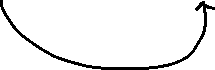
**Question 2.**

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 sorts.

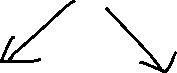


[10]

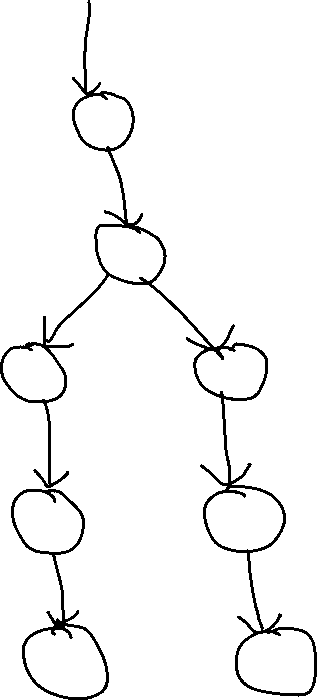
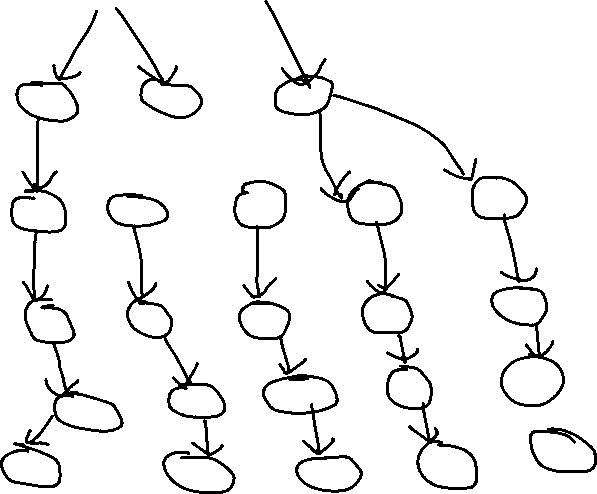
There are several topological sorts that can be obtained from this graph. One such topological sort is provided below using the source removal algorithm. The source removal algorithm looks for a source node(one without incoming edge) then removes it first and reduces the count of the incoming edges or incident edges leading out of this node.



Students must show the way the counts are reduced upon each removal of a node



****



**There are 7 in total topological sort for the above tree as can be seen from the above topological trees.**

**Question 3**

Write recursive function called reduce which takes a parameter 'x' (a positive integer, >=0 ), and reduce it to '0' recursively.

[6 ]

Algortihm reduce(int n)

//recursive algorithm to reduce an integer n to 0

if (n <= 0)

return n;

else

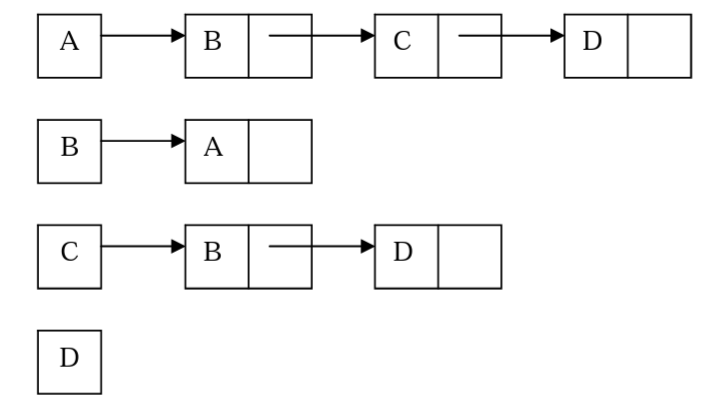
return reduce(n-1);

endif

EndAlg

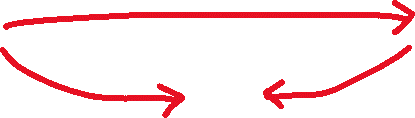
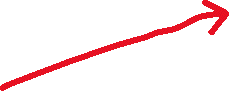
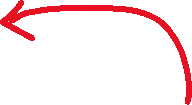
**Question 4**

Below is an adjacency matrix for a directed graph.





1. Draw a picture of the graph that has the above adjacency list representation.



1. Draw an equivalent adjacency matrix for the same graph.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **A** | 0 | 1 | 1 | 1 |
| **B** | 1 | 0 | 0 | 0 |
| **C** | 0 | 1 | 0 | 1 |
| **D** | 0 | 0 | 0 | 0 |

1. Is the graph a DAG? Explain your answer.

The graph is NOT a DAG (directed acyclic graph) because there is a cycle in the graph from A to B and from B to A.

[3,3,3=9]

**Question 5**

Below is a certain divide and conquer algorithm that is trying to sort the given array.

1. What is the name of this divide and conquer algorithm?

**MergerSort**

1. Provide the remaining steps/iterations so that the array becomes sorted.
2. Wha is the time complexity for this algorithm.

O(n.log n)

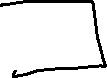
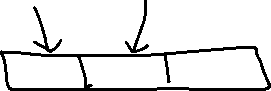
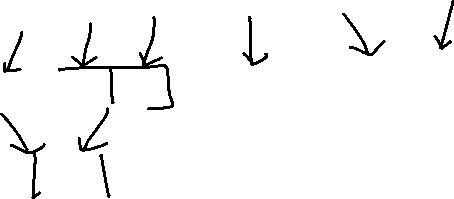
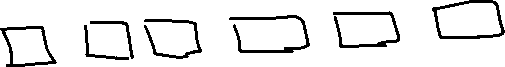
1. what is the space complexity for this algorithm

O(n)

**A diagram of a diagram

Description automatically generated**

[2, 3 = 5]



**Question 6**

Parts 6a and 6b both refer to the following sequence of integers:

7, 2, 5, 8, 6, 1, 4

6a) Sort the sequence from smallest to largest using selection.

sort. Show each step on a new line, underline the sorted

part of the array.

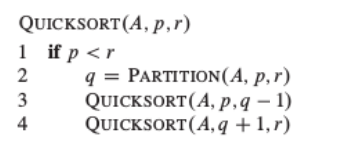
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 2 | 5 | 8 | 6 | 1 | 4 |
| **1** | 2 | 5 | 8 | 6 | 7 | 4 |
| **1** | **2** | 5 | 8 | 6 | 7 | 4 |
| **1** | **2** | **4** | 8 | 6 | 7 | 5 |
| **1** | **2** | **4** | **5** | 6 | 7 | 8 |
| **1** | **2** | **4** | **5** | **6** | 7 | 8 |
| **1** | **2** | **4** | **5** | **6** | **7** | 8 |
| **1** | **2** | **4** | **5** | **6** | **7** | **8** |

Sorted now

6b) Sort the sequence from smallest to largest using quicksort. Show each step on a new line.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 2 | 5 | 8 | 6 | 1 | 4 |
| **2** | 7 | 5 | 8 | 6 | 1 | 4 |
| **2** | **1** | 5 | 8 | 6 | 7 | 4 |
| **2** | **1** | **4** | 8 | 6 | 7 | 5 |
| **1** | **2** | **4** | **5** | 6 | 7 | 8 |
| **1** | **2** | **4** | **5** | 6 | 7 | 8 |
| **1** | **2** | **4** | **5** | **6** | **7** | **8** |
| **1** | **2** | **4** | **5** | **6** | **7** | **8** |
| **1** | **2** | **4** | **5** | **6** | **7** | **8** |
| **1** | **2** | **4** | **5** | **6** | **7** | **8** |

You may use the quicksort and partition methods below for 6b.

 A math equations on a white background

Description automatically generated

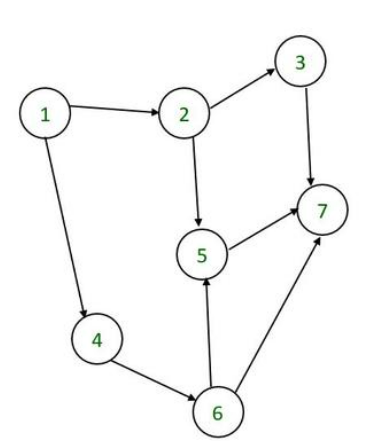
Qalkoasdjojdqosjik

hkhk

[5,5 =10]

**Question 7**

1. Using the DFS algorithm, trace the way the algorithm will visit the nodes in the following graph(assuming node 1 is the start node) , showing how they were put on the stack and how they were removed. Show all traces ( no mark will be given if you just draw or show the final answer) . *Let them start from node 1.*

****



|  |
| --- |
|  |
| **6** |
| **4** |
| **3** |
| **7** |
| **5** |
| **2** |
| **1** |

**Output : 7, 5, 3, 2, 6, 4, 1**

1. Prove by induction that 12+32+52+ 72+…+(2n-1)2 = 1/3n(4n2-1) for n>= 1, n ∈ N.

TO PROVE:

12+32+52...+(2n−1)2=3n(2n−1)(2n+1)​ ∀n∈N

PROOF:

P(n)=12+32+52...+(2n−1)2=3n(2n−1)(2n+1)​

P(1): (2×1−1) 2 = = 1 = 1\*(2−1)(2+1)/3​

⇒ (1) 2 = 1 = 1×1×3/3 ​= 1

∴ L.H.S=R.H.S (Proved)

∴P(1) is true.

Assumption :

Now, let P(k) is true.

Then, P(k)=12+32+52...+(2k−1) 2=3k(2k−1)(2k+1)​

Now, we have to prove that P(k+1) is also true.

P(k+1)=12+32+52...+(2k−1) 2+2(k+1)−1] 2

=P(k)+(2k+2−1) 2

=P(k)+(2k+1)2

=3k(2k−1)(2k+1)​+(2k+1) 2

 =3k(2k−1)(2k+1)+3(2k+1) 2​

=3(2k+1)[k(2k−1)+3(2k+1)]​

=3(2k+1)[2k2−k+6k+3]​

=3(2k+1)[2k2+5k+3]​

=3(2k+1)[2k2+2k+3k+3]​

=3(2k+1)[2k(k+1)+3(k+1)]​

=3(2k+1)(2k+3)(k+1)​

=3(k+1)[2(k+1)+1][2(k+1)−1]​

∴p(k+1) is also true (Proved)

[5, 10 = 15]

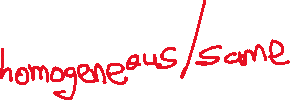
**Question 8**. Fill in the blanks.



a. The operation of processing each element in a tree is known as \_\_\_\_\_\_\_\_\_\_\_\_\_/tree traversal \_



b. Arrays are best data structure for relatively \_\_\_\_\_\_\_\_\_\_\_\_\_ collections of data



c. The best case occurs in linear search algorithm when an item is somewhere in the \_beginning \_\_\_ of the array

d. Algorithm efficiency in terms of time is measured by counting the number of \_\_\_\_\_\_\_\_\_



e. Algorithms can be measured in machine independent way using the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ model.



[1,1,1,1,1=5]

**Question 9.** Write a pseudo code for the binary search algorithm. Argue that worst case running time of binary search is O (lg n)

Accept either recursive or iterative, since the question did not specify

public static int binarySearch(int[] nums, int target)

    {

        int left = 0, right = nums.length - 1;

         while (left <= right)

        {

            int mid = (left + right) / 2;

             if (target == nums[mid]) {

                return mid;

            }

             else if (target < nums[mid]) {

                right = mid - 1;

            }

             else {

                left = mid + 1;

            }

        }

        return -1;

    }

[6]

Recursive version

int binarySearchRecursive(int nums[], int low, int high, int target)

{

    if (low > high) {

        return -1;

    }

    int mid = (low + high)/2;

    if (target == nums[mid]) {

        return mid;

    }

  else if (target < nums[mid]) {

        return binarySearchRecursive (nums, low, mid - 1, target);

    }

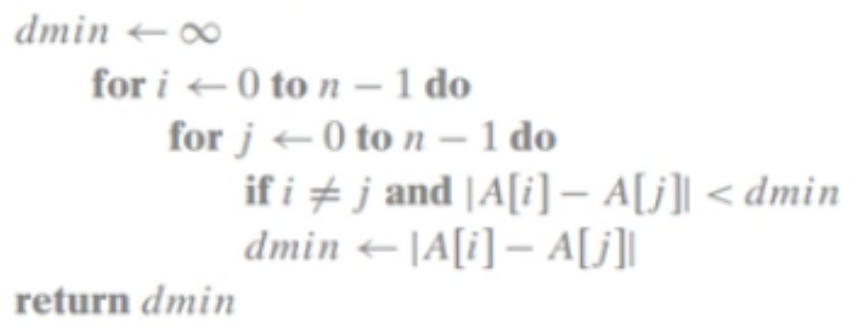
   else {

        return binarySearchRecursive (nums, mid + 1, high, target);

    }

}

**Question 10.** Deduce the time complexity of the following piece of code



fragment. Show all workings. Partial mark will be given to those who only quote the time complexity without showing the workings.

[5]

