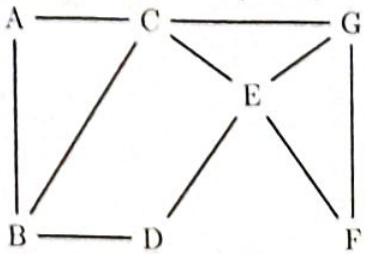
Fall 2018: CSE 221 | Discrete Mathematics | Akdeniz University

Tuesday 08/01/2019 | Final Exam | Duration: 90 minutes

Name: Solutions | Student No:

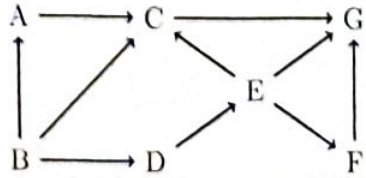
P1 [16 (1*16) points] Graph Definitions - Undirected Graph



Tick **ALL** appropriate definitions for each sequence:

Sequence	Walk	Path	Circuit	Cycle
A-C-E-F-G-E-C-B	✓			
A-C-E-F-G-E-D-B-A	✓		✓	
B-C-G-F-E-D-B	✓		✓	✓
A-C-G-E-D	✓	✓		

P2 [24 (2*8+8*1) points] Graph Basics - Directed Graph



Write the in-degrees of the vertices: A: 1 B: 0 C: 3 D: 1

Write the out-degrees of the vertices: E: 3 F: 1 G: 0

Is there a cycle in this graph? Yes · (No)

Give a topological order for the graph: B D E A C F G

✓ Other possible solutions are also accepted. All arrows must go to right.

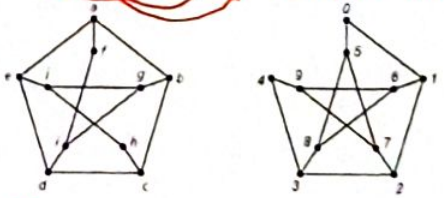
P3 [20 (10*2) points] Graph Isomorphism

a) Are the graphs on the right isomorphic? If yes give an isomorphism, if not explain why.

a	b	c	d	e	f	g	h	i	j
5	7	9	6	8	0	2	4	1	3

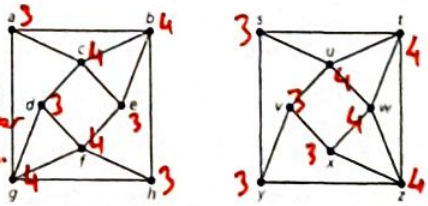
Yes ✓

or 9 7 5 8 6 4 2 0 3 1

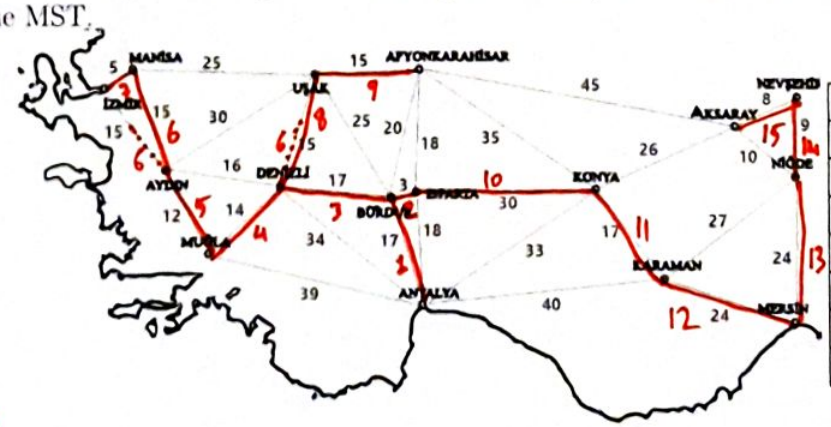


b) Are the graphs on the right isomorphic? If yes give an isomorphism, if not explain why. No. Because the 1st graph does not have 3-degree neighbor vertices while the 2nd one has.

a	b	c	d	e	f	g	h



P4 [20 (4*1+1*16) points] Minimum Spanning Trees In the map below, find a minimal spanning tree by using Prim's Algorithm starting from Antalya and write the cities in the order you add them to the MST.



1	Antalya	9	Uşak
2	Burdur	10	Afyon
3	Isparta	11	Konya
4	Denizli	12	Karaman
5	Muğla	13	Mersin
6	Aydın	14	Niğde
7	Manisa	15	Nevşehir
8	İzmir	16	Aksaray

Also possible:
7 İzmir
8 Manisa
9 Uşak
10 Afyon

or

7 Uşak	7 Uşak
8 İzmir	8 Manisa
9 Manisa	9 İzmir
10 Afyon	10 Afyon

or

7 U	7 U
8 A	8 A
9 M	9 İ
10 İ	10 M

P2 Extra

B→D→E→F→G, A→C

A and C can be anywhere as long as A→C, B→A, E→C, C→G points towards right.

P5 [20 (10*2) points] Graph Programming Suppose that you already have a graph class and a method that returns the shortest path between two given vertices. (Assume all edges have weight 1.) Your task is to add two new methods to this class, getEccentricity() and getDiameter(). Fill these methods below. Recall that eccentricity of a vertex is the distance to the farthest vertex and diameter is the longest distance in a graph.

```
public class Graph {
    ArrayList<Vertex> vertices;
    public Graph() { /*Already written*/ }
    ...
    public int shortestPath(Vertex v, Vertex w) { /*Already written*/ }
    public int getEccentricity(Vertex v) { //Fill this method
        int max = 0;
        for (int i=0; i< vertices.size(); i++){
            int temp = shortestPath(v, vertices.get(i));
            if (temp > max)
                max = temp;
        }
        return max;
    }
    public int getDiameter() { //Fill this method
        int max = 0;
        for (int i=0; i<vertices.size(); i++) {
            int temp = getEccentricity(vertices.get(i));
            if (temp > max)
                max = temp;
    }   }
        return max;
}
```

PBonus [10 points] Graph Programming Suppose you also have a method that returns the neighbor vertices of a vertex. Also suppose that vertices have a boolean field visited which is false by default. Write a method that starts from a vertex and visits all unvisited vertices exactly once. (just change visited field of each vertex to true.)

```
public ArrayList<Vertex> getNeighbors(Vertex v) { /*Already written*/ }
public void visitAll(Vertex v) { //Fill this method
    if (v.visited == false) {
        v.visited = true;
        for (int i=0; i< getNeighbors(v).size(); i++) {
            visitAll(getNeighbors(v).get(i));
        }
}
```