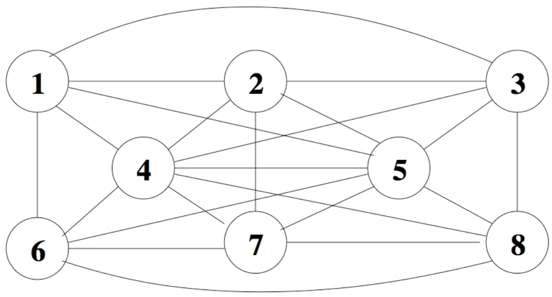
# Foundations of Computer Science (Semester 2) – 2015

## Assessed Exercise Sheet 9 – 10% of Continuous Assessment Mark

## Deadline : 12noon Sunday 22nd March, via Canvas

**Question 1 (22 marks)**

Represent the following undirected graph as an adjacency matrix:




| | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* |
|---|---|---|---|---|---|---|---|---|
| *1* | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| *2* | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| *3* | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| *4* | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| *5* | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| *6* | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| *7* | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| *8* | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

Show how depth-first traversal of the graph can be performed starting from vertex 1. Write down the stack and visited vertices at each stage.

There is no need to add items to the stack that have already been visited, so we have:

| Stack | Visited vertices |
|---|---|
| 1 | |
| 2, 3, 4, 5, 6, ~~1~~ | 1 |
| 3, 4, 5, 7, ~~2~~, 3, 4, 5, 6, ~~1~~ | 1, 2 |
| 4, 5, 8, ~~3~~, 4, 5, 7, ~~2~~, 3, 4, 5, 6, ~~1~~ | 1, 2, 3 |
| 5, 6, 7, 8, ~~4~~, 5, 8, ~~3~~, 4, 5, 7, ~~2~~, 3, 4, 5, 6, ~~1~~ | 1, 2, 3, 4 |
| 6, 7, 8, ~~5~~, 6, 7, 8, ~~4~~, 5, 8, ~~3~~, 4, 5, 7, ~~2~~, 3, 4, 5, 6, ~~1~~ | 1, 2, 3, 4, 5 |
| 7, 8, ~~6~~, 7, 8, ~~5~~, 6, 7, 8, ~~4~~, 5, 8, ~~3~~, 4, 5, 7, ~~2~~, 3, 4, 5, 6, ~~1~~ | 1, 2, 3, 4, 5, 6 |
| 8, ~~7~~, 8, ~~6~~, 7, 8, ~~5~~, 6, 7, 8, ~~4~~, 5, 8, ~~3~~, 4, 5, 7, ~~2~~, 3, 4, 5, 6, ~~1~~ | 1, 2, 3, 4, 5, 6, 7 |
| ~~8~~, ~~7~~, 8, ~~6~~, 7, 8, ~~5~~, 6, 7, 8, ~~4~~, 5, 8, ~~3~~, 4, 5, 7, ~~2~~, 3, 4, 5, 6, ~~1~~ | 1, 2, 3, 4, 5, 6, 7, 8 |

(There are other possible orderings since neighbours are unordered.)

Show how breadth-first traversal of the graph can be performed starting from vertex 1. Write down the queue and visited vertices at each stage.

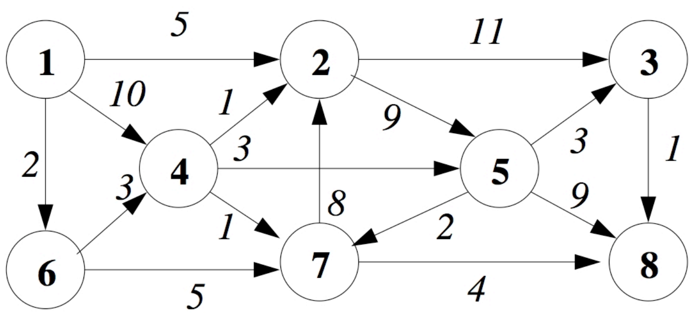
In this case there is no need to add items to the queue that are already in it or have already been visited, so we have:

| Queue | Visited vertices |
|---|---|
| 1 | |
| ~~1~~, 2, 3, 4, 5, 6 | 1 |
| ~~1, 2~~, 3, 4, 5, 6, 7 | 1, 2 |
| ~~1, 2, 3~~, 4, 5, 6, 7, 8 | 1, 2, 3 |
| ~~1, 2, 3, 4~~, 5, 6, 7, 8 | 1, 2, 3, 4 |
| ~~1, 2, 3, 4, 5~~, 6, 7, 8 | 1, 2, 3, 4, 5 |
| ~~1, 2, 3, 4, 5, 6~~, 7, 8 | 1, 2, 3, 4, 5, 6 |
| ~~1, 2, 3, 4, 5, 6, 7~~, 8 | 1, 2, 3, 4, 5, 6, 7 |
| ~~1, 2, 3, 4, 5, 6, 7, 8~~ | 1, 2, 3, 4, 5, 6, 7, 8 |

(There are other possible orderings since neighbours are unordered.)

**Question 2 (24 marks)**

Represent the following directed graph as a weight matrix:




| | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* |
|---|---|---|---|---|---|---|---|---|
| *1* | 0 | 5 | ∞ | 10 | ∞ | 2 | ∞ | ∞ |
| *2* | ∞ | 0 | 11 | ∞ | 9 | ∞ | ∞ | ∞ |
| *3* | ∞ | ∞ | 0 | ∞ | ∞ | ∞ | ∞ | 1 |
| *4* | ∞ | 1 | ∞ | 0 | 3 | ∞ | 1 | ∞ |
| *5* | ∞ | ∞ | 3 | ∞ | 0 | ∞ | 2 | 9 |
| *6* | ∞ | ∞ | ∞ | 3 | ∞ | 0 | 5 | ∞ |
| *7* | ∞ | 8 | ∞ | ∞ | ∞ | ∞ | 0 | 4 |
| *8* | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 0 |

Represent the same graph as an array of adjacency lists.

[1] → [2, 5] → [4, 10] → [6, 2]
[2] → [3, 11] → [5, 9]
[3] → [8, 11]
[4] → [2, 1] → [5, 3] → [7, 1]
[5] → [3, 3] → [7, 2] → [8, 9]
[6] → [4, 3] → [7, 5]

[7] → [2, 8] → [8, 4]
[8] →

Perform a breadth-first traversal of the graph starting at vertex 1. Explain the process you used.

The graph is traversed by adding neighbours to a queue, but one can ignore any vertices that are already in the queue or have already been visited, so we have:

Queue: ~~1, 2, 4, 6, 3, 5, 7, 8~~
Traversal: 1, 2, 4, 6, 3, 5, 7, 8

**The following sub-question was inadvertently left out of the question sheet. Obviously, marks will not be lost for not doing it, but it is included here for your information.**

Perform a depth-first traversal of the graph starting at vertex 1. Explain the process you used.

The graph is traversed by adding neighbours to a stack, but one can ignore any vertices that have already been visited, so we have:

Stack: ~~7~~, ~~8~~, ~~3~~, ~~5~~, ~~2~~, ~~4, 6~~, ~~1~~
Traversal: 1, 2, 3, 8, 5, 7, 4, 6

**Question 3 (16 marks)**

What is the computational complexity of breadth-first and depth-first traversal in terms of the number of vertices $v$ and the number of edges $e$ when the graph is represented as an adjacency matrix?

The complexity of both forms of traversal is $O(v^2)$ because every entry in the $v \times v$ adjacency matrix must be checked, regardless of whether a link exists.

How do those computational complexities change if the graph is instead represented as an adjacency linked list?

In this case, the complexity of both forms of traversal will be $O(v + e)$ because there are $v$ linked lists and $e$ links to follow.

**Question 4 (14 marks)**

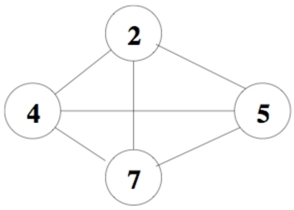
An undirected graph is said to be *connected* if and only if for every pair of non-identical vertices there exists a path from one vertex to the other. Explain in words how you could determine whether a given graph is connected.

Perform either breadth-first or depth-first graph traversal from any vertex, and count the number of vertices visited. If that number equals the number of vertices in the graph, the graph is connected. Otherwise, it is not connected.

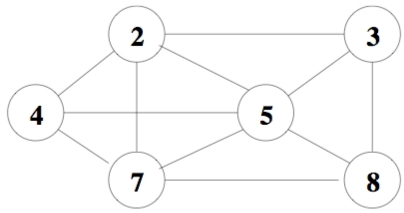
**Question 5  (24 marks)**

What property must a graph satisfy to be called *planar?* Determine, without using any theorems concerning $K_5$ and $K_{3,3}$, which of the following graphs are planar, and which are not? In each case, explain in detail how you arrived at your answer.

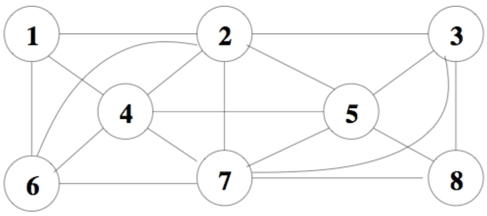
(a)



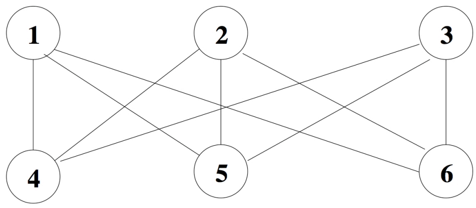

(b)




(c)



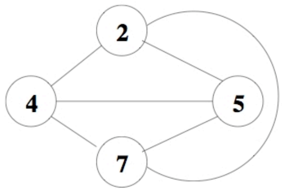

(d)



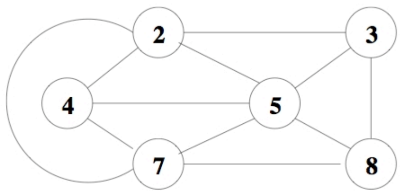

A graph is said to be *planar* if all its edges can be drawn in a two-dimensional plane with no edges crossing each other.

Graphs (a), (b) and (c) are planar because they can be drawn without crossing edges as follows:

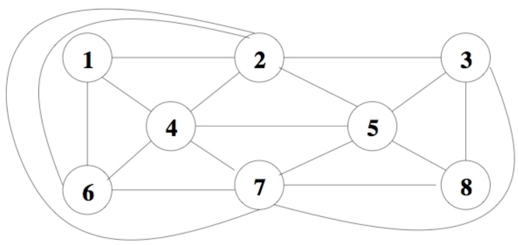
(a)



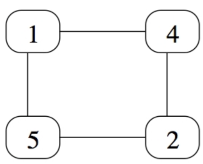

(b)



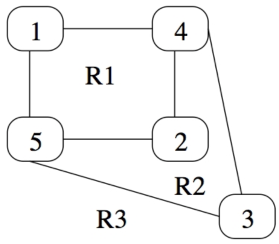

(c)

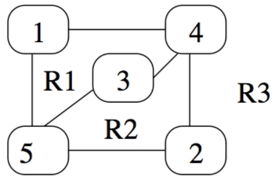



For graph (d) one can explore systematically all possible ways of drawing the graph, and conclude that they will all involve at least one pair of edges crossing, so it is not planar. For example, vertices 1, 4, 2 and 5 form a circular sub-graph:




If we then add vertex 3 it can either be placed inside or outside that circle:




Either way leaves only three distinct regions R1, R2 and R3. If we then add vertex 6, whichever region we put it in leaves it unable to connect to at least one of vertices 1, 2 and 3 without crossing one of the existing edges. Therefore the graph is not planar.