Graphs: More practice problems

1. The degree requirements for a major can be modeled as a *DAG* (directed acyclic graph), where vertices are required courses and an edge (x, y) means course x must be completed prior to beginning course y. Let's consider a hypothetical Computer science major with the following assumptions:

   - You must take **all** CS classes listed in the requirements.
   - All prerequisites must be obeyed (no instructor override).
   - There is a course, csci101, that must be taken before any other course by everyone
   - Every course is offered every semester (unlike our department).
   - There is no limit to the number of courses you can take in one semester, and, you are guranteed to get into any CS course you register for (again, unlike our department).

   Describe an efficient algorithm to compute the minimum number of semesters required to complete the degree and analyze its running time. Aim for $O(|V| + |E|)$ time.

   We expect: pseudocode, justification, analysis.

2. You are given an image as a two-dimensional array of size $m \times n$. Each cell of the array represents a pixel in the image, and contains a number that represents the color of that pixel (for e.g. using the RGB model).

   A segment in the image is a set of pixels that have the same color and are **connected**: each pixel in the segment an be reached from any other pixel in the segment by a sequence of moves up, down, left or right.

   Design an efficient algorithm to find the size of the largest segment in the image.

3. An undirected graph is called *bipartite* if the set of vertices can be partitioned into two disjoint sets, such that all edges in G have one endpoint in each set (put differently, such that no edge in G has both endpoints in the same set). Describe an algorithm which, given an undirected graph $G = (V, E)$, determines whether it is bipartite.

   Hint: Your algorithm should use that a graph is bipartite if it is 2-colorable (and the other way around). A graph G is called *k-colorable* if it is possible that the vertices can be assigned one of k different colors, in such a way that no edge connects vertices of the same color. Use BFS and try to 2-color the vertices. If you succeed, the graph is 2-colorable and therefore bipartite.

   We expect: pseudocode, justification and run time analysis.