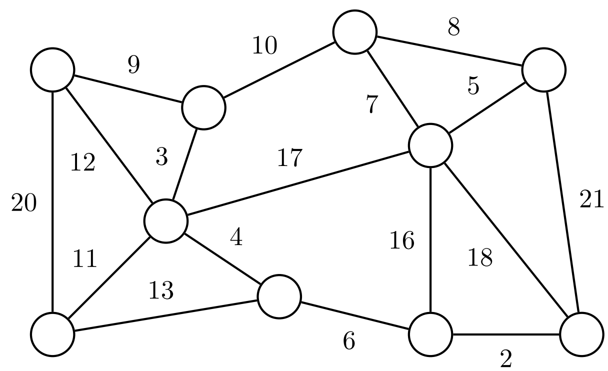
# CS 199: Homework 3 - Revised Version

Due by 10:00 a.m. on Monday, Apr 2

1. **Minimum Spanning Trees:**

   (a) The figure below shows a graph, where the length of each edge is indicated by the number next to that edge. Find a Minimum Spanning Tree of this graph, by using the procedure/algorithm discussed in class. Recall that our algorithm was to always select the smallest-weight edge remaining unless it introduces a cycle.

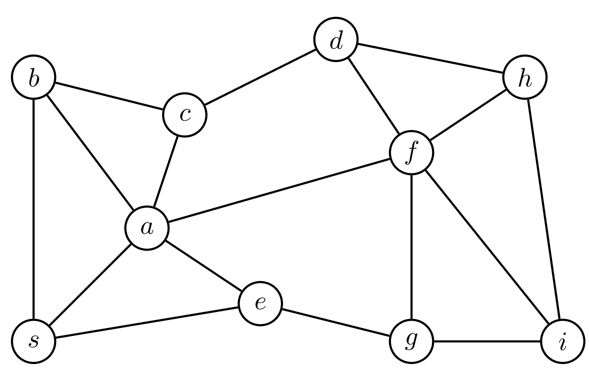



   (b) Suppose, given some graph with edge lengths, you have found the minimum spanning tree of the graph. Now suppose the length of each edge doubles. (That is, if an edge had length $\ell$ before, it now has length $2\ell$.) Will the minimum spanning tree of the graph with the new edge lengths always look the same as the minimum spanning tree of the old graph (that is, will it contain the same edges), or is it possible that the structure of the tree will change? Your answer should be "the same" or "it may change".

   (c) Suppose, as in part (b), that you have found the minimum spanning tree of a graph, and then suppose that the length of each edge is squared. That is, if an edge had length $\ell$ before, it now has length $\ell^2$. Does the minimum spanning tree change?

2. **Breadth-First Search:**

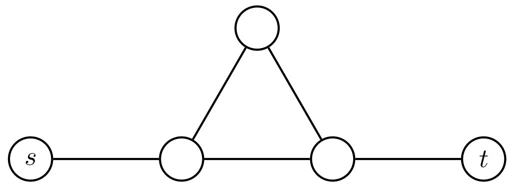
   (a) The figure below shows a graph. Suppose we were to run a breadth-first search starting from the vertex[1] named $s$. In what order are the other vertices discovered? (If there is more than one possible order, you can list any one.)




[1]A *vertex* is another name for a node. The plural of vertex is vertices, so *vertices* means exactly the same thing as *nodes*

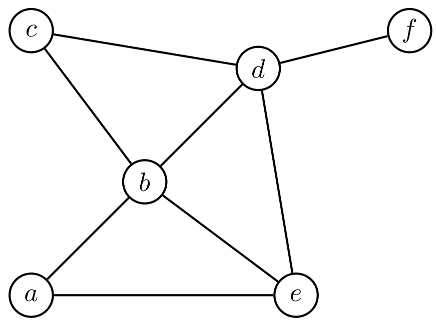
(b) Describe a method/algorithm which, given a graph $G$ and a vertex $s$, finds the vertex of $G$ that is furthest from $s$ (that is, requires the most 'hops' to reach). Your algorithm should also output the number of hops required to reach the furthest vertex.

(c) The *diameter* of a graph is defined as the distance between the pair of vertices in it that are farthest apart. (Think of the diameter of a circle. Also, here, when we say 'distance', we mean the shortest way to get from one to the other.) In the graph below, the diameter is 3, because the shortest path from $s$ to $t$ uses 3 hops; for any other pair of vertices, the distance between them is at most 2.




For the next graph, the table below shows the pairwise (shortest) distance between every pair of vertices.

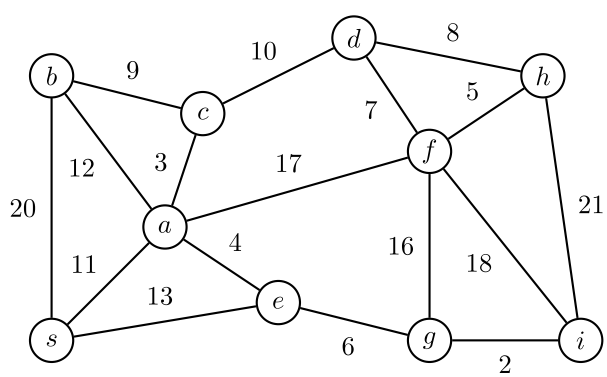
| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | - | 1 | 2 | 2 | 2 | 3 |
| b | 1 | - | 1 | 1 | 1 | 2 |
| c | 2 | 1 | - | 1 | 2 | 2 |
| d | 2 | 1 | 1 | - | 1 | 1 |
| e | 2 | 1 | 2 | 1 | - | 2 |
| f | 3 | 2 | 2 | 1 | 2 | - |




Vertices $a$ and $f$ are the pair farthest from each other; the shortest distance from $a$ to $f$ is 3. Therefore, the diameter of the graph is 3.

Describe a general method that, given a graph $G$, finds its diameter.
*(HINT: Use your algorithm from part (b))*

3. For the graph below (with edge lengths written next to the edges), write in a table the shortest distance from $s$ to each other vertex. You should find these shortest distances using Dijkstra's shortest-path algorithm, described in class. Write out the contents of the set $S$ at each step of Dijkstra's algorithm.



4. **Coloring:** Recall that in the graph coloring problem, we want to use as few colors as possible so that every vertex of the graph can get some color, and no two vertices that are joined by an edge get the same color. Explain why each of the following two problems is really a type of graph coloring problem. That is, describe how you could use a graph coloring algorithm to solve these problems.

   - You are employed by a zoo that is facing budget cuts, and is trying to find the fewest different habitats in which it can keep all its animals safely. We could try to use one habitat for *all* the animals, but if the lions and deer are together, the deer won't survive for very long. For every pair of animals, you know whether that pair can live in the same habitat or not. Why is finding the minimum number of habitats the same as a graph coloring problem?
   - Suppose you had a map of the United States, and you want to color each state with some color, but to make the map easier to use, you want to be sure that no two states that share a border have the same color. How many colors do you need?

     More generally, in the map-coloring problem, we are given any map (of states, countries, counties, whatever), and we need to color the different regions of the map, so that if two regions share a border, they get different colors. How is finding the minimum number of colors you need to color the map the same as a graph coloring problem? [2]

5. **Python programming:** *Do Not Panic!* The next question will have you write a python program. If you spend some time thinking about it and get stuck, or are not sure where to start, ask us for help. Please do not try to find solutions on the internet, and do not ask any of your friends who may know how to program.

   It is completely ok not to get the program exactly right; even though lab next week is not focused on python/programming, we will spend a litle bit of time working on this program. You should email us your solution before then, though.

   Write a python program that asks the user to pick a number between 0 and 16 (but not to enter it into the computer), and then tries to find out what the number is by asking the user questions of the form "Is your number equal to 1?", "Is your number equal to 2?", etc. The user must answer only "yes" or "no", and the program cannot ask any other types of questions, such as "Is your number bigger than 10?", for example. Once the program finds out what the number is, it should print "The number you guessed is", followed by the number. (So, for instance, if the user guessed 3, once the program finds that out, it should say "The number you guessed is 3.")

[2] A famous question, dating back to 1852, asks whether every map can be colored using at most 4 colors. In 1976, Kenneth Appel and Wolfgang Haken, at the University of Illinois, solved it by proving that 4 colors are always enough, though sometimes we can manage with 2 or 3. Interestingly, this was the first major mathematical proof that required the use of a computer.