Discussion Prompt Questions
Module 6
Brian Loughran
Johns Hopkins Data Structures

1. What attributes of a problem might you consider when deciding how to represent a graph?
2. What are some advantages of using a linked structure to represent a graph? Disadvantages?
3. Why might an array be better (or worse) than a linked structure to represent a graph?
4. Can you name some applications for which a graph representation might be useful?
5. What aspects of graphs make them a good (or poor) match for recursive solutions?

ANSWERS:

1. In deciding how to represent a graph for a particular problem one may consider if there is a need to represent as a directed graph. If all the connections are two-way, then it would be prudent to represent as an undirected graph, as that will reduce the amount of edges in the graph. If some connections are only one-way (e.g. links between Wikipedia pages, a representation of roads that include some one-way roads, etc.) then it may be more useful to represent as a directed graph. You may also consider if there are any loops in the graph, or if the graph is entirely one directional. If the graph is one directional, it may make sense to represent as a tree, with parents and children nodes to make traversing the graph more intuitive.
2. An advantage to using the linked implementation for representing a graph is that the graph is naturally represented and allocated in memory using the linked implementation. A disadvantage of using the linked implementation is that it may be difficult to determine the size of the graph based on the linked implementation, unless you keep track of some metadata as you add and delete nodes.
3. An array could potentially be better than a linked implementation if you have to "start" at different places in the graph often, and don't want to traverse the linked structure to find the starting point. Accessing the starting point in an array would be very easy. A disadvantage to using an array implementation is the fact that the array can run out of space and produce an overflow. Another disadvantage is that the graph could be too sparsely populated, and allocating a lot of memory could be a waste.
4. Google maps represents the roads as a graph. This is a rather large graph, but you could imagine each intersection being a node, and each road being an edge. Then, you can take the length of the road divided by the speed limit of the road to get a time, and set that as an edge length for each edge. If Google detects some traffic on an edge (road) they could increase the edge length, therefore approximating the amount of time to travel on the road. If traffic becomes so bad on a road that is no longer the shortest path, Google can recommend another path by searching the graph.
5. Graphs are good for recursive solutions because traversing a graph is an inherently repetitive process. Each time you reach a node, you should be looking for paths to the next node. If you do not want to include loops, end cases are simple to create, an end case is when you either reach your destination, or you have no other paths to explore. However, if you want to include loops in your analysis, recursion may not be a great option, as you would either get stuck in an infinite loop or your end case would be more complicated.