Module 7 - Discussion Prompt Questions
Adam Rich
March 20, 2018
Section EN.605.202.87.SP18

1. What attributes of a problem might you consider when deciding how to represent a graph?

   Is there something in the data that could be represented as points or vertices?
   Are there relationships between different points? (edges)
   Are those edges weighted or do they simple imply adjacency?
   Whether a vertex can be linked to just one other vertex or mutiple vertices.
   Are vertices ever going to point to themselves?

2. What are some advantages of using a linked structure to represent a graph? Disadvantages?

   A graph is a linked structure, logically, so it makes sense that this would be a way to implement a graph. Linked structures allocate space dynamically, so for very large graphs this might be useful (to reduce space complexity). Depending on the operations that need to be done on this graph a linked structure might help (or hurt!). A disadvantage might be traversing the graph. Or, figuring out how to represent that nodes can have a varying number of edges.

3. Can you name some applications for which a graph representation might be useful?

   Optimizing paths between related objects or sets of objects, like figuring out the shortest path between two cities given a set of roads and intermediate points. Or storing data in a non-sorted way but still maintaining relationships like in a binary sort tree.

4. What aspects of graphs make them a good (or poor) match for recursive solutions?

   A node only knows about the nodes it is adjacent to, so if you need to operate on a whole bunch of related nodes joined by paths then a graph naturally breaks up the problem into smaller, recursively oriented chunks. You will run in to problems if the graph is cyclical or if there are nodes that do not have a path between them. Also, there will be a lot of duplicated effort, as in many recursive solutions.