

## Quiz #5 Answers

1. What are the two methods we discussed for storing a graph in a program? Under what conditions do you want to use each one?

The two methods are adjacency lists and adjacency matrices. The adjacency matrix allows you to determine if there is an edge connecting two vertices in  $O(1)$  time, but it always requires  $O(V^2)$  memory. The adjacency list only requires  $O(E)$  memory and when the matrix is sparse,  $E$  is  $O(V)$ , then the access time isn't so bad. However, it becomes slow if the graph is dense.

2. Describe an algorithm for finding a minimum spanning tree. You can use pseudocode or plain english.

The basic algorithm is to repeatedly find "safe" edges and add them in. A safe edge is any edge that is part of some minimum spanning tree. To help in picking safe edges we saw a theorem that says that if we have a set of edges,  $A$ , and a cut respecting  $A$ , then the edge crossing the cut with lightest weight is safe to add to the set  $A$  to form a minimum spanning tree. The two specific algorithms we looked at to do this were Kruskal's and Prim's. In Kruskal's we repeatedly select the lowest weight edge in the entire graph that doesn't form a cycle with the edges already selected. Sorting the edges by weight makes the first part easy. Determining if an edge would form a cycle can be done quickly with a disjoint sets data structure. In Prim's algorithm we start with a particular vertex as our "tree". Then we repeatedly add the lightest edge that connects our current tree to a vertex not yet in the tree. To make this work efficiently we have to keep a priority queue of the edges coming out of the current tree.

Extra credit: What type of graph representation is used by most all-pairs, shortest path algorithms?

They use a matrix type representation.