Graphs

1. From Hilfinger Fall 2007:

Suppose you have some weighted undirected graph. This graph has three nodes of interest. In one node, there is a cat. In another there is a rat. In a third, there is a mousehole. Each "turn", the cat and mouse can move to a node connected to one they are currently on. If the mouse reaches the mousehole before the cat can, then he wins. If the cat reaches it first or they tie, the cat wins.

Describe how you would determine the winner in 2 sentences or less.
Dijkstra's Algorithm

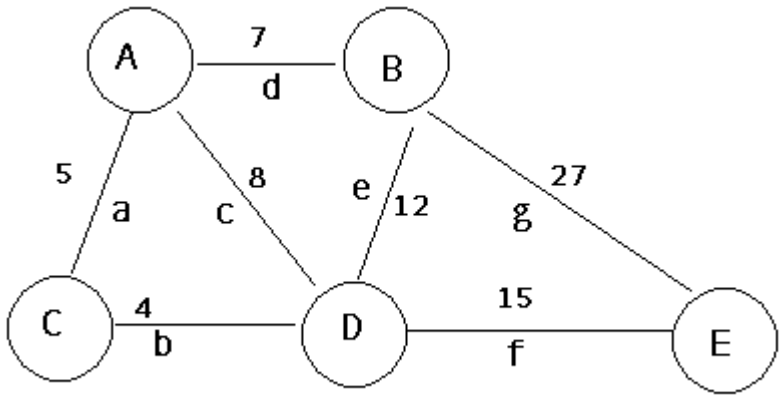
2. From Shewchuk Spring 2004:

a. How long does it take to determine if an undirected graph contains a vertex that is connected to no other vertices [i] if you use an adjacency matrix; [ii] if you use adjacency lists.
i: v^2, since you need to look at all the entries of the matrix
ii: v. You walk down this list, and if any of them is null, you have found such a vertex.
A -> Neighbors of A
B -> Neighbors of B.
C -> Null.

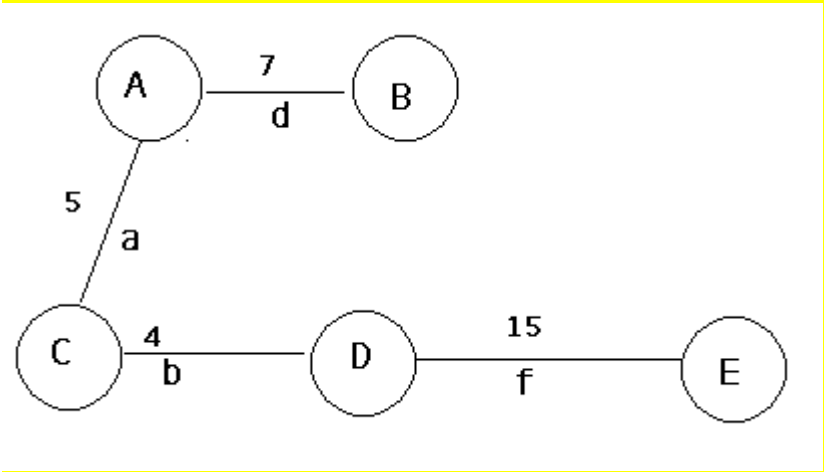
b. An undirected graph contains a "cycle" (i.e., loop) if there are two different simple paths by which we can get from one vertex to another. Using your favorite graph traversal algorithm, how can we tell if an undirected graph contains a cycle?
Mark all nodes traversed. If you ever encounter a marked node during traversal, then you've found one..

c. Recall that an undirected graph is "connected" if there is a path from any vertex to any other vertex. If an undirected graph is not connected, it has multiple connected components. A connected component" consists of all the vertices reachable from a given vertex, and the edges incident on those vertices. Suggest an algorithm based on DFS (possibly multiple invocations of DFS) that counts the number of connected components in a graph.
Start with a list of vertices. For the first vertex, run DFS, marking each vertex as you touch it. Go down the list of vertices, and start DFS again on each unmarked vertex. The number of times DFS was run is the number of components that exist.

3. Consider this graph.



a. Draw out the minimum spanning tree constructed by Kruskal's Algorithm.



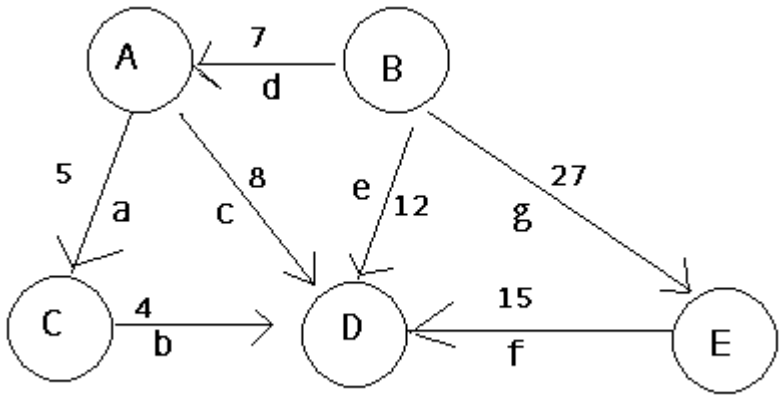
b. What order did you draw the edges in?
b a d f

c. Draw out the minimum spanning tree by Prim's Algorithm.
Same as before.

d. What order did you draw the vertices in, starting from A?
A C D B E

e. Were the two trees you drew different? Could they have been for some other graph? If so, draw that graph and two different possible minimum spanning trees.
Those two trees were not different. But another graph's could be. Start with an isoceles triangle. Remove any one of the equal edges.

f. Suppose the graph was directed, in this order.



After having done a topological sort, if you were to print them in a sorted order, what orders are possible that you printed?

B E A C D
B A E C D
B A C E D

Linear Sort

4. Apply the Radix sort algorithm on the following list of English words: COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX. Show the result after successive sorts on increasingly significant character position.
COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX
SEA TEA MOB TAB DOG RUG DIG BIG BAR EAR TAR COW ROW NOW BOX FOX
TAB BAR EAR TAR TEA SEA BIG DIG MOB DOG COW ROW NOW BOX FOX RUG
BAR BIG BOX COW DIG DOG EAR FOX MOB NOW ROW RUG SEA TAB TAR TEA

5. Before, we could consider that selection sort can be an especially bad instance of Quicksort (where your pivot is always the lowest). Can we apply some kind of relation to the following other sorts? IE, can we say one could be a special case of the other?

a. Bucket Sort v. Radix Sort
Repeated Bucket Sort is like a MSD Radix Sort.
b. Bucket Sort v. Counting Sort
Counting Sort is Bucket Sort with 1-value per bucket.
c. Radix Sort v. Counting Sort
No great answers.
d. Bucket Sort v. QuickSort.

3 Buckets, middle on is the pivot.

6.

a. Show the result of bucket sorting the following list into 10 buckets: 66, 4, 85, 93, 68, 76, 74, 39, 100, 17

0-9 4
10-19 17
20-29 39
30-39
40-49
50-59
60-69 66 68
70-79 76 74
80-89
90-100 93 100

b. In a bucket sort such as the one above, you may have ended up with more than one item per bucket. In such cases, you can do a bucket sort on each bucket to further sort the
elements. If we do two such operations, what is the total running time?
n to do the initial bucket sort, then for b buckets you need to do n/b numbers per bucket. Add them up.
O(n + b*(n/b)) = O(2n) = O(n)