CS61B Spring 2015 Final Section (Amanda, Brendan, Japheth)

Solutions will be available at: http://goo.gl/JEbEJS

Counting Sort and Radix Sort

(1) Radix Sort Practice. Use [least significant digit] radix sort to sort the following numbers: `32 75 17 55 52 57 2 10 51 30`. (Use a single digit for each pass of the sort.)

(2) Choosing a Radix. Suppose you are using radix sort to sort a list of Java ints, and it is important to sort them as quickly as possible. Explain why you would **never** choose to use exactly 512 buckets for this sort. (Source: Shewchuk Sp'05 Final Exam)

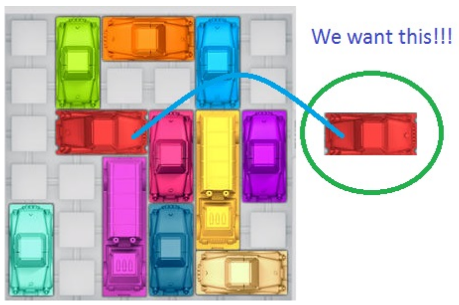
(3) Analysis. What is the runtime and memory complexity of the following sorting algorithms? Let *n* be the number of items to sort, and *q* be the number of items in the alphabet.

	Runtime Complexity	**Memory Complexity**
Counting Sort		
Radix Sort (LSD)		

Graph Traversals

(4) Family Tree Traversal. You are looking at your family tree, and and you want to traverse it with either DFS or BFS to find one particular person who is still alive. Which would you use?

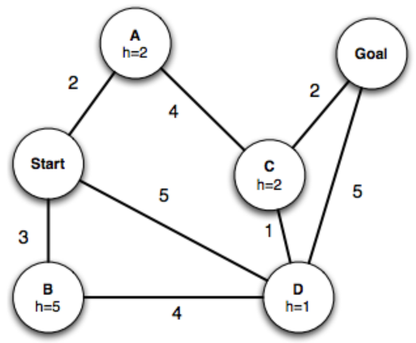
(5) Rush Hour. Say we are trying to write a program that solves rush hour in the least number of moves. We wish to move the red car out of the grid using some series of moves. Assume we already have data structures for our Grid (including a `move(Car, Move)` method) and an algorithm implemented that outputs the next possible moves: `ArrayList<Move> getNextMoves(Grid)`.



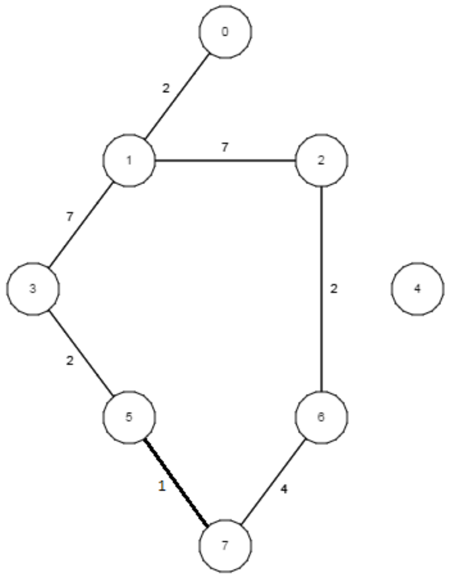
(a) How do we figure out which car we will end up moving next?

(b) What data structure will we need so that we do not have infinite loops of game state?

(6) A* Search. Perform A* search on the given graph, with the heuristic *h* and the edge weights to find the shortest path from the START to GOAL vertices. List the nodes in the order which we would visit them, as well as the path that A* search would return.



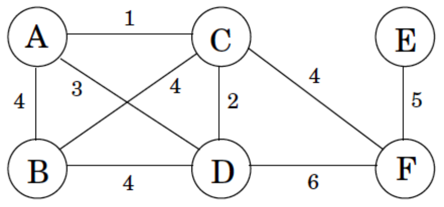
(7) Dijkstra's Algorithm. Using Dijkstra's on the graph to the right, compute the cost of going from vertex 0 to any vertex.



Minimum Spanning Trees and Tries

(8) Minimum Spanning Trees. Under what circumstances could a graph have multiple valid minimum spanning trees? One valid minimum spanning tree?

(9) Prim's and Kruskal's Algorithm. List the edges of the minimum spanning tree of the graph below, in the order which they are added to the MST. Do so for Prim's Algorithm, then for Kruskal's Algorithm. (For Prim's algorithm, start at node A. For both, use an arbitrary tiebreaking scheme.) What is the total weight of this MST?



(10) More MSTs. Suppose you have a weighted graph where all edge weights are positive and distinct. Is it possible for a tree of shortest paths from some vertex s and the minimum spanning tree of the graph to not share any edges? If so, give an example. If not, give a reason why.

(11) Tries. Prefix-free encoding is one where no code word is a prefix of another code word. Say we took a large array of words, compressed them using prefix-free encoding, then inserted them into a trie. What special property would this trie have? (Hint: What assumption can we now make about the nodes in this trie?)