CS140 - Sample Problems for Midterm 2

Below are some practice problems to help give you study for the upcoming midterm. Note that not all of these would necessarily be good exam problems, but are there to provide you with some additional practice on the materials.

1. A student wants to walk up a staircase with n steps. They can skip up to 2 steps at a time: in other words, they can go from step k to step $k+1$, they can skip one step and go from step k directly to step $k+2$, or they can skip 2 steps and go from step k directly to step $k+3$. Let $Num(i)$ be the number of ways that the student can get to step i. Write and justify a recursive formula for $Num(i)$. If you were to implement your solution as an iterative dynamic program, what would the pseudocode be? What are bounds on the space and time requirements of your dynamic programming algorithm?

2. Change revisited

   In class we discussed the change problem and, in particular, proved that a greedy strategy was optimal for US denominations (penny=1, nickel=5, dime=10, quarter=25).

   The change problem in general can be specified as: make change for an amount of money C with as few coins as possible for coin denominations with values $v_1 > v_2 > \ldots > v_n$ (all integers), where $v_n = 1$.

   (a) The greedy approach only works for certain coin values. Given an example of coin denominations and a target amount such that the greedy strategy does not provide the optimal solution.

   (b) Give a dynamic programming solution for the function which calculates the minimum number of coins necessary to make the amount. Make sure you give both a written description of what's stored in the dynamic programming table as well as the recursive definition.

   (c) What is the size of your dynamic programming table? What entry contains the answer? What is the running time of your algorithm?

   (d) Fill out the dynamic programming table assuming the denominations are $v_1 = 6, v_2 = 5, v_3 = 1$ and the value $C = 10$.

3. Suppose that you need to insert 100,000 keys into a hashtable with 200,000 buckets. You are very unlucky and all of the keys hash to the same bucket. Would you be better off using resolution by chaining, linear probing or double hashing? Explain your answer.

4. Suppose we are given a connected undirected graph with distinct edge weights. We greedily delete the heaviest edge that does not disconnect the graph, until we cannot delete any more

edges. Is the result guaranteed to be a minimum spanning tree of the original graph? If so, justify your answer. If not, give a counterexample.

5. T/F: There is no way to insert n elements into a hashtable of size m where $n > m$.

6. T/F: If two values x, y are different (i.e., $x \neq y$) then there is no way for them to have the same hash function value, i.e., $h(x) = h(y)$.

7. T/F: If we make all the edges in a DAG undirected, we will always end up with a tree (undirected, acyclic graph).

8. T/F: If we modify DFS and BFS to record the edge (u, v) that was traversed to get to a particular vertex v and we run them starting at a particular vertex r in a tree (undirected, acyclic graph), BFS and DFS would always record identical edges for all vertices.

9. T/F: If we modify DFS and BFS to record the edge (u, v) that was traversed to get to a particular vertex v and we run them starting at a particular vertex r on a DAG, BFS and DFS would always record identical edges for all vertices.