CS140 - Checkpoint 2 Sample Problems: SOLUTIONS

Below are some practice problems to help give you study for the upcoming checkpoint. 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. Staircase

   SOLUTION:

   $Num(1) = 1$
   $Num(2) = 2$
   $Num(3) = 4$
   $Num(i) = Num(i-1) + Num(i-2) + Num(i-3)$

   To get to step i, we can either take one step, which leaves us with $i-1$ stairs left, a bigger step (skipping one step) which leaves us at $i-2$ stairs left, or an even bigger step (skipping two steps), leaving $i-3$ stairs left.

   Nearly identical to Fibonacci, fill in the first three values and then from 4 up to n. $O(n)$ space and time.

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.

   $v_1 = 6, v_2 = 5, v_3 = 1$ and the value $C = 10$:)

   (b) DP solution

   Let $change(i)$ be the way to make change summing to i with the fewest counts. We can write this recursive by considering making change with each coin and then taking them min, i.e., for each v_j, the number of coins used is $1 + change(i - v_j)$ (only considering coins where $v_j <= i$). $change(0) = 0$ and the table would be filled in starting at 1 up to C.

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

SOLUTION: The size is C, the amount of change to be made. The answer is at entry n. The running time is $O(nC)$.

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

SOLUTION:

```
[0, 1, 2, 3, 4, 1, 1, 2, 3, 4, 2]
 0  1  2  3  4  5  6  7  8  9  10
```

3. Hashing

   Double hashing. Both chaining and linear probing would end up with a linear collection of things and linear run-time for lookup.

4. Yes! We only ever delete an edge that is part of a cycle, which would never be part of any MST.

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

   SOLUTION: False. If your table uses collision resolution by chaining n can be greater than 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., $h(x) = h(y)$.

   SOLUTION: False. This is one of the reasons collisions occur.

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

   SOLUTION: False. Consider the DAG with four vertices a, b, c, d with edges $a \to b$, $a \to c$, $b \to d$, $c \to d$. If these edges are made undirected, we have a cycle.

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.

   SOLUTION: True. Given a starting point, every vertex has a single parent. Regardless of how the graph is traversed from this starting point, a given vertex will always be explored from its parent.

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.

   SOLUTION: False. In a DAG, a vertex can have more than one parent. Depending on the DFS/BFS implementation and the order in which the adjacent vertices are traversed, the way a vertex if visited may vary.