CSE 101 Spring 2024 Homework 6 Due: Thursday 5/30 at 11:59pm

1. (10 points) Let G be an directed acyclic graph with vertex set V and edge set E.

   Design a (recursive) backtracking algorithm that counts the number of topological orderings of the vertices in V.

   (Hint: use an array indexed by vertices that keeps track of the indegree of each vertex.)

   [high level description or pseudocode of algorithm. Proof of correctness. (No runtime analysis necessary.)]

2. Recall the electric car problem but this time, each battery also has a price $p[i]$ to replace:

   > Suppose you are driving along a road in an electric car. The battery of the electric car can bring you $x[0]$ miles. There are battery stations along the way at positive positions $D[1], \dots D[n]$ (in sorted order.) Each battery station can *replace* your battery and give you a new battery that can bring you a certain number of miles. The distances of the batteries are given in the array $x[0], x[1], \dots, x[n-1]$ and the price of each battery to replace is $p[1], \dots, p[n-1]$.
   >
   > You wish to start at position 0 with a full battery and end at position $D[n]$ by replacing batteries with the minimum total cost.

   (a) (4 points)

   Recall the following greedy algorithm that worked in homework 4:

   > **Candidate Greedy Strategy III:**
   > Travel to the battery station with the largest $D[i]+x[i]$ value (in other words, the battery that can take you the farthest down the road.) Replace the battery at that station and repeat the process starting from that station until you can reach position $D[n]$ and then go directly there.

   Give a counterexample as to why this does not always give you the optimal solution.

   (b) (7 points)

   Design a *reduction* algorithm that uses Dijkstra's algorithm. Compute the time analysis in terms of the number of battery stations n.

   4 points for correct high level algorithm.
   3 points for runtime analysis.
   (No proof of correctness)
   (Your algorithm should run in $O(n^2)$ time.)

   (c) (10 points)

   Design a DP tabulation algorithm by ordering the subproblems from n to 0: (step 1 and 4 have been done for you)

   1: Define the subproblems:
      Let $G[k]$ be defined to be the minimum price it takes to get to battery station n assuming you start at battery station k with the battery from battery station k.
   2: Define and evaluate the base cases
   3: Establish the recurrence for the tabulation.
   4: Determine the order of subproblems:
      Order the subproblems from n to 0.
   5: Final form of output.
   6: Put it all together as pseudocode
   7: Runtime analysis

3. In the kingdom of Dynamoprogamia, a law was passed that the currency used in the Kingdom would be in the denominations of perfect square numbers (i.e. in denominations of 1, 4, 9, 16, 25 and so on). You can assume the denominations can go as large as you want it to go as long as it's a perfect square number. Let's call this currency Square-Dollars. A money lender in this Kingdom wants to lend money to his customers in such a way that he gives them the least number of coins possible. For instance, if the customer wants to borrow 8 Square-Dollars, the money lender can give it to him in multiple different ways: eight 1-dollar coins, Total number of coins: 8. Four 1-dollar coins and one 4-dollar coin, Total number of coins: $4 + 1 = 5$. Two 4-dollar coins, Total number of coins: 2 Hence the money lender will give the customer 2 coins of value 4 Square-Dollars each as that uses just 2 coins.

   (a) (4 points) The greedy strategy for this: "Pick the maximum denomination that's less than *or equal to* the amount you need to lend and subtract it from the amount you need to lend. Continue this process until you're left with nothing else to lend." Provide a counter example for this.

   (b) (10 points) Devise a Dynamic programming-based algorithm that returns the minimum number of coins needed to make change for n Square-Dollars. (your algorithm should run in $O(n^{1.5})$ time.)

      1: Define the subproblems:

         Let $SD[i]$ be defined to be the minimum number of coins needed to make change for i Square-Dollars.

      2: Define and evaluate the base cases

      3: Establish the recurrence for the tabulation.

      4: Determine the order of subproblems:

      5: Final form of output.

      6: Put it all together as pseudocode

      7: Runtime analysis

4. (12 points) You are given an $n \times n$ matrix of 0's and 1's: $(A_{i,j})_{1 \le i \le n,\ 1 \le j \le n}$

   Design a DP tabulation algorithm that finds the length of a side of the largest square entirely made of 1's: (step 1 has been done for you)

   1: Define the subproblems:

      Let $S[i, j]$ be defined to be the length of the side of the largest square entirely made of 1's with bottom right corner at entry $[i, j]$

   2: Define and evaluate the base cases

   3: Establish the recurrence for the tabulation.

   4: Determine the order of subproblems:

   5: Final form of output.

   6: Put it all together as pseudocode

   7: Runtime analysis