Homework 4

COMP221 Spring 2025 - Suhas Arehalli

Complete the problems below. Check the course website & syllabus for further instructions.

If any problem is unclear, or you think you found a typo, please let me know ASAP so I can clarify!

Problems

1. **Making Change**

   Suppose you are a cashier making change for a customer. Consider the following problem description for the problem of providing change with the fewest number of coins:

   PROBLEM (Minimal Change Set)**.**
   Input: $k \in \mathbb{Z}$, a number of cents, and $C = \{c_1, \dots c_n\} \subseteq \mathbb{Z}_+$.

   Output: A multi-set (set that allows duplicates) L such that

   - $\forall c \in L, c \in C$
   - $\sum_{c \in L} c = k$
   - $|L|$ is minimized.

   if such a set exists. $NULL$ otherwise.

   That is, I need you to find the fewest possible coins such that the values of those coins add up to k.

   You might have seen in Discrete Math that a greedy strategy works for US coin denominations (i.e., the case where $C = \{1, 5, 10, 25\}$. In fact, you can prove this correct using a clever, but tedious exchange argument.

   (a) Prove via counterexample that the greedy strategy will not work for all sets of coin denominations C. That is, construct C such that the greedy approach fails to find the minimal choice of coins.

   (b) Describe the solution to this problem in terms of a recurrence relation. For simplicity, write a solution that finds the **size** of the optimal set rather than constructing the set itself.

   ***HINT**:* Think carefully about how to frame the problem as a sequential decision process. It may be helpful to think about a way to avoid the mistake a greedy algorithm makes on your counterexample in the prior part, or to consider the subset sum problem.

(c) Provide pseudocode for a dynamic programming algorithm that computes the **minimum number of coins** needed to make k units of change that works for *any* coin system. For full points, provide a solution that runs in $O(k|C|)$ time and $O(k)$ space. A solution with a worse space complexity can still get the majority of points.

2. **Degree-Constrained Spanning Trees** *(Based on Skiena 11-12)*

(a) The low-degree spanning tree problem (LOW-DEGREE-SPAN) is a decision problem defined as follows:

PROBLEM (LOW-DEGREE-SPAN)**.**
Input: An undirected graph $G = (V, E)$ and $k \in \mathbb{Z}_{\geq 0}$
Output: TRUE iff there exists a spanning tree where each vertex in the tree has degree $\leq k$. FALSE otherwise.

Prove (using a Karp reduction) that LOW-DEGREE-SPAN is NP-hard. Make sure you provide both a description of the reduction and a brief justification of the correctness of the reduction as part of your answer.
****Hint**:** Skim through the NP-hard problems we've seen so far and see which ones seem most similar. Then try and construct a reduction in the appropriate direction.

(b) Consider the *high-degree spanning tree problem* (HIGH-DEGREE-SPAN) which is defined as follows:

PROBLEM (HIGH-DEGREE-SPAN)**.**
Input: An undirected graph $G = (V, E)$ and $k \in \mathbb{Z}_{\geq 0}$.
Output: TRUE if G contains a spanning tree which contains a vertex with degree $\geq k$ (within the spanning tree). FALSE otherwise.

Show that HIGH-DEGREE SPAN is in P by providing pseudocode for a polynomial-time algorithm to solve this problem.
****Hint****: Consider methods we've seen before that have you construct spanning trees in a graph!