Section 8: Final Review

1. Reduction

Consider the following problems:

HAM-PATH **Input**: A **directed** graph G **Output**: True if there is a Hamiltonian Path in G, that is a path that visits each vertex **exactly** once.
HAM-CYCLE **Input**: A **directed** graph G **Output**: True if there is a Hamiltonian Cycle in G, that is, a path $v_0, v_1, ..., v_n$ that visits each vertex **exactly** once along with the edge (v_n, v_0).

Suppose that HAM-PATH is NP-hard. Use that fact to show HAM-CYCLE is NP-hard.

2. Max-Flow/Min-Cut

A group of traders are leaving Switzerland, and need to convert their Francs (the local currency) into various international currencies. There are n traders and m currencies. Trader i has T_i Francs to convert. The bank has B_j Francs worth of currency j. Trader i is willing to trade as much as C_{ij} of his Francs for currency j. (For example, a trader with 1000 Francs might be willing to convert up to 700 of his Francs for USD, up to 500 of his Francs for Japaneses Yen, and up to 500 of his Francs for Euros).

Assuming that all traders give their requests to the bank at the same time, describe an algorithm that the bank can use to satisfy the requests (if it can).

3. DP Non-Adjacent LCS

The sequence $C = c_1, ..., c_k$ is a *non-adjacent subsequence* of $A = a_1, ..., a_n$, if C can be formed by selecting non-adjacent elements of A, i.e., if $c_1 = a_{r_1}, c_2 = a_{r_2}, ..., c_k = a_{r_k}$, where $r_j < r_{j+1} - 1$. The non-adjacent LCS problem is given sequences A and B, find a maximum length sequence C which is a non-adjacent subsequence of both A and B.

This problem can be solved with dynamic programming. Give a recurrence that is the basis for a dynamic programming algorithm. You should also give the appropriate base cases, and explain why your recurrence is correct.

4. DP Electoral College

The problem is to determine the set of states with the smallest total population that can provide the votes to win the electoral college. Formally, the problem is:

Let p_i be the population of state i, and v_i the number of electoral votes for state i. All electoral votes of a state go to a single candidate, so the winning candidate is the one who receives at least V electoral votes, where $V = \lfloor (\sum_i v_i)/2 \rfloor + 1$. Our goal is to find a set of states S that minimizes the value of $\sum_{i \in S} p_i$ subject to the constraint that $\sum_{i \in S} v_i \geq V$.

(a) The dynamic programming solution for this problem involves computing a function OPT where OPT(i, v) gives the minimum populations of a set of states from $1, 2, ..., i$ such that their votes sum to exactly v. Give a recursive definition of OPT and an explanation as to why it is correct.

(b) What are the base cases for your function OPT.