CS 473 ✧ Fall 2024
Homework 8

Due Tuesday, November 12, 2024 at 9pm Central Time

---

1. Given an undirected graph $G = (V, E)$ with three special vertices u, v, and w, describe and analyze an algorithm to determine whether there is a simple path from u to w that passes through v.

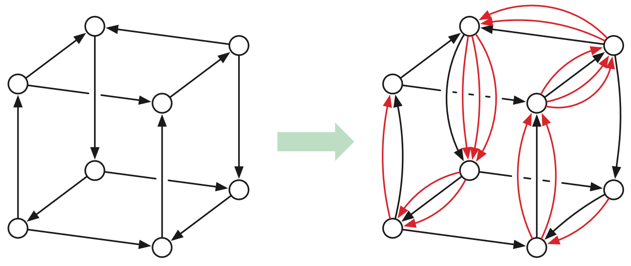
   [Hint: Do **not** *try to modify a standard graph traversal algorithm like DFS or BFS. This problem is on this homework for a reason.]*

2. An *Euler tour* in a directed graph G is a closed walk (starting and ending at the same vertex) that traverses every edge in G exactly once; a directed graph is *Eulerian* if it has an Euler tour. Euler tours are named after Leonhard Euler, who was the first person to systematically study them, starting with the Bridges of Königsberg puzzle.

   (a) Prove that a directed graph G with no isolated vertices is Eulerian if and only if (1) G is strongly connected—for any two vertices u and v, there is a directed walk in G from u to v and a directed walk in G from v to u—and (2) the in-degree of each vertex of G is equal to its out-degree. *[Hint: Flow decomposition!]*

   (b) **[Extra credit[1]]** Suppose that we are given a strongly connected directed graph G with no isolated vertices that is *not* Eulerian, and we want to make G Eulerian by duplicating existing edges. Each edge e has a duplication cost $€(e) \geq 0$. We are allowed to add as many copies of an existing edge e as we like, but we must pay $€(e)$ for each new copy. On the other hand, if G does not already have an edge from vertex u to vertex v, we cannot add a new edge from u to v.

   Describe an algorithm that computes the minimum-cost set of edge-duplications that makes G Eulerian.



Making a directed cube graph Eulerian.

---

[1]The intended solution for this problem uses techniques that we will not cover this semester, but which are described in additional lecture notes on flows with edge demands and vertex balances and minimum-cost flows.

3. Suppose we are given an array $A[1..m][1..n]$ of non-negative real numbers. We want to ***round*** A to an integer matrix, by replacing each entry x in A with either $\lfloor x \rfloor$ or $\lceil x \rceil$, without changing the sum of entries in any row or column of A. For example:

$$\begin{bmatrix} 1.2 & 3.4 & 2.4 \\ 3.9 & 4.0 & 2.1 \\ 7.9 & 1.6 & 0.5 \end{bmatrix} \longmapsto \begin{bmatrix} 1 & 4 & 2 \\ 4 & 4 & 2 \\ 8 & 1 & 1 \end{bmatrix}$$

   (a) Describe and analyze an efficient algorithm that either rounds A in this fashion, or correctly reports that no such rounding is possible.

   (b) Prove that a legal rounding is possible *if and only if* the sum of entries in each row is an integer, and the sum of entries in each column is an integer. In other words, prove that either your algorithm from part (a) returns a legal rounding, or a legal rounding is *obviously* impossible.