1 Realizable Degree Sequences (50 points)

Recall that the in-degree of a vertex \( v \) in a directed graph is the number of edges that go into \( v \), i.e. the number of edges of the form \((u, v)\). Similarly, the out-degree is the number of edges leaving \( v \), i.e. the number of edges of the form \((v, u)\). Let \( d_{v,\text{in}} \) denote the in-degree of \( v \), and let \( d_{v,\text{out}} \) denote the out-degree. In this problem we will not allow any loops (edges of the form \((u, u)\)) or multiedges (i.e. any edge \((u, v)\) can only be in the graph at most once, although both \((u, v)\) and \((v, u)\) could be in the graph).

(a) (15 points) Suppose you are given the following set of degree constraints for a four-node directed graph:

\[
\begin{align*}
d_{1,\text{in}} &= 0, \quad d_{1,\text{out}} = 2 \\
d_{2,\text{in}} &= 1, \quad d_{2,\text{out}} = 2 \\
d_{3,\text{in}} &= 1, \quad d_{3,\text{out}} = 1 \\
d_{4,\text{in}} &= 3, \quad d_{4,\text{out}} = 0
\end{align*}
\]

Is there a directed graph, with no multi-edges or self-loops, that satisfies these constraints? If so, what is it? If not, why not?

(b) (35 points) This type of specification, where we are given the in- and out-degrees of every vertex, is called a degree sequence. The question above was whether a particular degree sequence is realizable – that is, whether there exists a graph having those degrees.

Give a polynomial-time algorithm that, given a degree sequence, will determine if it is realizable, and if it is will produce a directed graph with those degrees. The graph should not have any self loops or multi-edges (i.e. for each directed pair \((i, j)\) with \(i \neq j\) there can be at most one edge from \(i\) to \(j\), although it is fine if there is also an edge from \(j\) to \(i\)). Prove that your algorithm is correct, and runs in polynomial time.

Hint: think flow, and think about creating a useful auxiliary graph.

2 Linear Programming (50 points)

Let’s consider linear programming formulations of the minimum spanning tree problem. We now have an (undirected) graph \( G = (V, E) \) and weights \( w : E \rightarrow \mathbb{R}^+ \). As we discussed in class, one way
of phrasing the minimum spanning tree problem is as finding the minimum cost connected subgraph which spans all nodes. This interpretation naturally gives rise to a straightforward LP relaxation which requires every cut to have at least one edge crossing it (fractionally). More formally, suppose that we have a variable $x_e$ for every edge $e$, and consider the following linear program. For all $S \subset V$, let $E(S, \bar{S})$ denote the edges with exactly one endpoint in $S$ and exactly one endpoint not in $S$.

$$\min \sum_{e \in E} w(e)x_e$$

subject to

$$\sum_{e \in E(S, \bar{S})} x_e \geq 1 \quad \forall S \subset V : S \neq \emptyset$$

$$x_e \geq 0 \quad \forall e \in E$$

Note that there are an exponential number of constraints, but let’s not worry about that.

(a) (25 points) Prove that the optimal value of this LP is at most the weight of the minimum spanning tree.

(b) (25 points) Unlike the examples in class, this is not an exact formulation of the MST problem. Find a graph $G = (V, E)$ and weights $w : E \rightarrow \mathbb{R}^+$ such that the optimal LP value is strictly less than the weight of the MST (and prove this).