This week, we’ll continue exploring NP-COMPLETE decision problems, and develop approximation algorithms for related versions of those problems.

1. **Traveling Salesman Problem.** In this problem, a salesman travels the country making sales pitches. The salesman must visit \( n \) cities and then return to her home city, all while doing so as cheaply as possible.

The input is a complete graph \( G = (V, E) \) along with nonnegative edge costs \( \{c_e : e \in E\} \). A tour is a simple cycle \((v_j_1, \ldots, v_j_n, v_j_1)\) that visits every vertex exactly once.\(^1\) The goal is to output the minimum-cost tour.

For many TSP applications (such as when the cost is proportional to the distance between two cities), it makes sense for the edges to obey the triangle inequality: for every \( i, j, k \), we have

\[
c_{(ik)} \leq c_{(ij)} + c_{(jk)}.
\]

This version is often called **Metric-TSP**.

The (decision version of the) Traveling Salesman Problem is NP-COMPLETE. For this problem, you will develop a 2-approximation algorithm for Metric-TSP.

(a) First, to gain some intuition, consider the following graph:

(b) *On your own* try to identify a cheap tour of the graph.

(c) Build some more intuition by computing the minimum spanning tree (MST) of the graph. Let \( T \) be your minimum spanning tree.

(d) Let OPT be the cheapest tour. Show that its cost is bounded below by the cost of the MST: \( \text{cost}(T) \leq \text{cost}(OPT) \).

\(^1\text{except for the start vertex, which we visit again to complete the cycle}\)
(e) Give an algorithm which returns a tour \( A \) which costs at most twice the cost of the MST: \( \text{cost}(A) \leq 2 \text{cost}(T) \).

(f) Conclude that your algorithm is a 2-approximation for \textsc{Metric-TSP}.

2. \textbf{Toy-Storage}. William has lots of toys of all different sizes. You’d like to purchase a number of bins in which to store the toys. Approximately how many bins will you need?

Let’s formalize the \textsc{Toy-Storage} problem as follows. Suppose there are \( n \) toys, with sizes \( s_1, \ldots, s_n \), with \( 0 < s_i < 1 \) for all \( i \). Assume each bin has size 1 and can hold any collection of toys whose total size is less than or equal to 1.

In this problem, you’ll develop a greedy approximation algorithm, which works by taking each toy in turn and placing it into the first bin that can hold it. Let \( S = \sum_{i=1}^{n} s_i \).

(a) Show that the optimal number of bins is at least \( \lceil S \rceil \).
(b) Show that the greedy algorithm leaves at most one bin half full.
(c) Prove that the number of bins used by the greedy algorithm is at most \( \lceil 2S \rceil \).
(d) Prove that the greedy algorithm is a 2-approximation algorithm for the \textsc{Toy-Storage} problem.