Exercises

1. Given an edge-weighted undirected connected chain-graph $G = (V, E)$, all vertices having degree 2, except two endpoints which have degree 1 (there is no cycle). Design an algorithm that preprocesses the graph in linear time and can return the distance of the shortest path between any two vertices in constant time (i.e., the $\mathcal{O}(|V|)$ preprocessing enables return of the shortest-path distance between any two vertices in $\mathcal{O}(1)$ time).

2. The interval scheduling problem is to find a largest compatible set - a set of non-overlapping intervals of maximum size. Suppose that instead of always selecting the first activity to finish, we select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

3. Let T be an MST of a weighted, undirected graph G. Given a connected subgraph H of G, show that $T \cap H$ is contained in some MST of H.

4. Given an edge-weighted undirected graph $G = (V, E)$, such that $|E| > |V|$ and all **edge-weights are distinct**.
   We define a second-best minimum spanning tree as follows. Let $\mathcal{T}$ be the set of all spanning trees of G, and let T' be a MST of G. Then a second-best minimum spanning tree is a spanning tree T such that $W(T) = \min_{T'' \in \mathcal{T} - \{T'\}}(W(T''))$, $W(\cdot)$ denotes weight of spanning tree.

   (a) Show that the minimum spanning tree is unique, but that the second-best minimum spanning tree need not be unique.

   (b) Let T' be the minimum spanning tree of G. Prove that G contains edges $(u, v) \in T'$ and $(x, y) \notin T'$ such that T' - $\{(u, v)\} \cup \{(x, y)\}$ is a second-best minimum spanning tree of G.

   (c) Let T be a spanning tree of G and for any two vertices u, $v \in V$, let $\max(u, v)$ denote an edge of maximum weight on the unique simple path between u and v in T. Design an $\mathcal{O}(|V|^2)$ time algorithm that, given T, computes $\max(u, v)$; for all u, $v \in V$.

   (d) Design an efficient algorithm to compute the second-best minimum spanning tree of G.

5. Given a graph $G = (V, E)$, a subset $S \subseteq V$ of vertices is said to be a *vertex cover* of G if for every edge $(u, v) \in E$, at least one of u, v belongs to the subset S. A minimum vertex cover of G is a vertex cover with minimum cardinality (i.e., smallest vertex cover).
   Design an algorithm to find a minimum vertex cover of a given tree $T = (V, E)$. Justify why your algorithm works, give pseudocode, and give an analysis of runtime complexity.

6. Given an edge-weighted directed graph $G = (V, E)$ such that all the edge-weights are positive. Let s and t be two vertices in G and $k \leq |V|$ be an integer. Design an algorithm to find the shortest path from s to t that contains exactly k edges.
   Note that the path need not be simple, and is permitted to visit vertices and edges multiple times.

7. You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \ldots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance a_n), which is your destination.

   You'd ideally like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel x miles during a day, the penalty for that day is $(200 - x)^2$. You want to plan your trip so as to minimize the total penalty- that is, the sum, over all travel days, of the daily penalties. Give an efficient algorithm that determines the optimal sequence of hotels at which to stop.