7. tutorial

1. Still connected. Design a linear-time algorithm for the following task. You are given an undirected graph $G = (V, E)$. Decide whether there is an edge $e \in E$ such that $G - e$ is still connected.

2. Prescribed Edges. Say we are given a subset of edges $F \subseteq E$ of a graph $G = (V, E)$. Does there always exist a spanning tree T such that $F \subseteq T$? (Try to first think about the case when F contains exactly one edge.)
How to find the lightest spanning tree T among those with $F \subseteq T$, for example by modifying Jarník's algorithm?

3. MST properties. The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated.

(1) If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.
(2) If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.
(3) Let e be any edge of minimum weight in G. Then e must be part of some MST.
(4) If the lightest edge in a graph is unique, then it must be part of every MST.
(5) If e is part of some MST of G, then it must be a lightest edge across some cut of G.
(6) If G has a cycle with a unique lightest edge e, then e must be part of every MST.
(7) Jarník's algorithm works correctly when there are negative edges.
(8) The shortest-path tree computed by Dijkstra's algorithm is necessarily an MST.
(9) The shortest path between two nodes is necessarily part of some MST.
(10) (For any $r > 0$, define an r-path to be a path whose edges all have weight $< r$.) If G contains an r-path from node s to t, then every MST of G must also contain an r-path from node s to node t.

4. MSTs in multigraphs. Consider how to modify one of the three algorithms for MST to work on multigraphs – that is, graphs which may contain loops and multiple edges between two vertices.

5. MSTs in planar graphs. How to find an MST of a planar graph in $\mathcal{O}(n)$?

6. Greedy and Knapsack. The KNAPSACK problem is the following[1]: we are given a collection of n items of sizes $a_1, \ldots, a_n \in \mathbb{N}$ and a capacity $K \in \mathbb{N}$, and the task is to find the largest subset of items which fits in the knapsack, that is, whose total size is at most K.
The algorithms for MST which we have described are all an example of *greedy algorithms*, that is, at each iteration, they make a choice which is best *in the moment*, without any regard for the future (will this block off the path to some better solution?). This greedy approach is a heuristic which can be applied to many problems (at each iteration, make a step which gives the most improvement *in this moment*), and in some cases, it can be proven that it approximates the optimum well. However, MST is one of the rare problems where the greedy approach optimally solves the problem.
Show that a greedy approach will not work for KNAPSACK. (This is not an exactly-defined question: there are several natural ways to define "a greedy approach to KNAPSACK". Explore the ones which come to you.)

7. Changing one. Let a graph G and its MST T be given. Now change the weight of one of the edges of G, and call the new MST T' (potentially $T = T'$). Describe the relationship of T, T' (how do they differ etc.).

[1] Actually, normally the items also have values, and we are trying to maximize the total value of packed items. So the formulation above is equivalent to saying that the value of each item is its size.