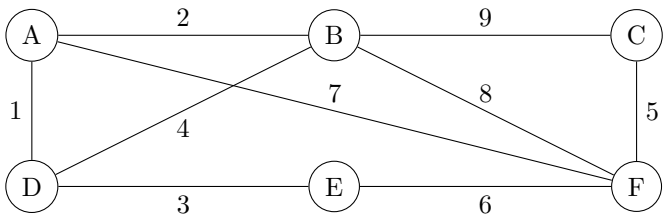
Exercises

Exercises should be completed **on your own.**

1. **(2 pt.)** Consider the graph G below.



   (a) **(1 pt.)** What MST does Prim's algorithm return? In what order does Prim's algorithm add edges to the MST when started from vertex C?

   (b) **(1 pt.)** What MST does Kruskal's algorithm return? In what order does Kruskal's algorithm add edges to the MST?

   [We are expecting: For both, just a list of edges. No justification is required.]

2. **(4 pt.)** At a Thanksgiving dinner, there are n food items $f_0, \ldots, f_{n-1}$. Each food item has a tastiness $t_i > 0$ (measured in units of deliciousness per ounce) and a quantity $q_i > 0$ (measured in ounces). There are q_i ounces of food f_i available to you, and for any real number $x \in [0, q_i]$, the total deliciousness that you derive from eating x ounces of food f_i is $x \cdot t_i$. (Notice that x here doesn't have to be an integer).

   Unfortunately, you only have capacity for Q ounces of food in your belly, and you would like to maximize deliciousness subject to this constraint. Assume that there is more food than you can possibly eat: $\sum_i q_i \geq Q$.

   (a) **(2 pt.)** Design a greedy algorithm which takes as input the tuples (f_i, t_i, q_i), and outputs tuples (f_i, x_i) so that $0 \leq x_i \leq q_i$, $\sum_i x_i \leq Q$, and $\sum_i x_i t_i$ is as large as possible. Your algorithm should take time $O(n \log(n))$.

   [We are expecting: Pseudocode and a short English explanation.]

   (b) **(2 pt.)** Fill in the inductive step below to prove that your algorithm is correct.

   - **Inductive hypothesis:** After making the t'th greedy choice, there is an optimal solution that extends the solution that the algorithm has constructed so far.
   - **Base case:** Any optimal solution extends the empty solution, so the inductive hypothesis holds for $t = 0$.
   - **Inductive step:** *(you fill in)*
   - **Conclusion:** At the end of the algorithm, the algorithm returns an set S^* of tuples (f_i, x_i) so that $\sum_i x_i = Q$. Thus, there is no solution extending S^* other than S^* itself. Thus, the inductive hypothesis implies that S^* is optimal.

   [We are expecting: A proof of the inductive step: assuming the inductive hypothesis holds for $t-1$, prove that it holds for t.]

Problems

You may talk with your fellow CS161-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

---

1. **(6 pt.)** Consider the problem of **making change.** Suppose that coins come in denominations $P = \{p_0, \ldots, p_m\}$ cents (for example, in the US, this would be $P = \{1, 5, 10, 25\}$, corresponding to pennies, nickels, dimes, and quarters). Given n cents (where n is a non-negative integer), you would like to find the way to represent n using the fewest coins possible. For example, in the US system, 55 cents is minimally represented using three coins, two quarters and a nickel.

   (a) **(3 pt.)** Suppose that the denominations are $P = \{1, 10, 25\}$ (aka, the US ran out of nickels). Your friend uses the following greedy strategy for making change:

Algorithm 1: makeChange(n)

Input: n and P
coins = [ ];
while $n > 0$ **do**
    $p^* \leftarrow \max\{p \in P : p \leq n\}$;
    $n = n - p^*$;
    coins.append(p^*) ;
return *coins*

   Your friend acknowledges that this won't work for general P (for example if $P = \{2\}$ then we simply can't make any odd n), but claims that for this particular P it does work. That is, your friend claims that this algorithm will always return a way to make n out of the denominations in P with the fewest coins possible.

   Is your friend correct for $P = \{1, 10, 25\}$?

   [We are expecting: Your answer, and either a proof or a counterexample. If you do a proof by induction, make sure to explicitly state your inductive hypothesis, base case, inductive step, and conclusion.]

   (b) **(3 pt.)** Your friend says that additionally their algorithm should work for any P of the form $P = \{1, 2, 4, 8, \ldots, 2^s\}$.

   Is your friend correct for $P = \{1, 2, 4, \ldots, 2^s\}$?

   [We are expecting: Your answer, and either a proof or a counterexample. If you do a proof by induction, make sure to explicitly state your inductive hypothesis, base case, inductive step, and conclusion.]

2. **(7 pt.)** On Thanksgiving day, you arrive on an island with n turkeys. You've already had thanksgiving dinner (and maybe you prefer tofurkey anyway), so you don't want to eat the turkeys; but you do want to wish them all a Happy Thanksgiving. However, the turkeys each have very different sleep schedules. Turkey i is awake only in a single closed interval $[a_i, b_i]$. Your plan is to stand in the center of the island and say loudly "Happy Thanksgiving!" at certain times $t_1, \ldots, t_m$. Any turkey who is awake at one of the times t_j will hear the message. It's okay if a turkey hears the message more than once, but you want to be sure that every turkey hears the message at least once.

   (a) **(3 pt.)** Design a greedy algorithm which takes as input the list of intervals $[a_i, b_i]$ and outputs a list of times $t_1, \ldots, t_m$ so that m is as small as possible and so that every turkey hears the message at least once. Your algorithm should run in time $O(n \log(n))$.
   [We are expecting: Pseudocode and an English description of the main idea of your algorithm, as well as a short justification of the running time.]

   (b) **(4 pt.)** Prove that your algorithm is correct.
   [We are expecting: A proof by induction]

3. **(6 pt.)** Let G be a connected weighted undirected graph. In class, we defined a minimum spanning tree of G as a spanning tree T of G which minimizes the quantity

$$X = \sum_{e \in T} w_e,$$

   where the sum is over all the edges in T, and w_e is the weight of edge e. Define a "minimum-maximum spanning tree" to be a spanning tree that minimizes the quantity

$$Y = \max_{e \in T} w_e.$$

   That is, a minimum-maximum spanning tree has the smallest maximum edge weight out of all possible spanning trees.

   (a) **(4 pt.)** Prove that a minimum spanning tree in a connected weighted undirected graph G is always a minimum-maximum spanning tree for G.
   [HINT: Suppose toward a contradiction that T is an MST but not a minimum-maximum spanning tree, and say that T' is a minimum-maximum spanning tree. Let (u, v) be the heaviest edge in T. Then deleting (u, v) from T will disconnect it into two trees, A and B. Now use A and B and T' to come up with a cheaper MST than T (and hence a contradiction).]
   [We are expecting: A proof.]

   (b) **(2 pt.)** Show that the converse to part (a) is not true. That is, a minimum-maximum spanning tree is not necessarily a minimum spanning tree.
   [We are expecting: A counter-example, with an explanation of why it is a counter-example.]

   (c) **(1 bonus pt.)** Give a deterministic $O(m)$ algorithm to find a minimum-maximum spanning tree in a connected weighted undirected graph G with n vertices and m edges.
   [We are expecting: Pseudocode, along with an English description of the idea of the algorithm, and an informal justification of correctness and running time.]