Precept Outline
- Defining Trees
- Binomial Heaps
- Algorithm Design

A. What is a tree, exactly?

Prove *formally* that the following conditions on an n-vertex undirected graph G are equivalent:

1. G is acyclic and connected;
2. G is maximal among all n-vertex acyclic graphs (i.e., adding an edge creates a cycle);
3. G is minimal among all n-vertex connected graphs (i.e., removing any edge disconnects it).

(Therefore, the mathematical definition of a tree is any – equivalently, all – of the above.)

B. Binomial Heaps

It's been a while since we discussed data structures for priority queues, but they'll make a comeback in the course! Since there's also a beautiful application of (so-called soft) heaps for MST algorithms, now is a great time to explore advanced priority queue implementations.

In order to construct *binomial heaps* (one such implementation that performs efficient *melding* – see below), we need to (recursively) define binomial *trees*:

- B_0 is a single node.
- B_k is two copies of B_{k-1} with an additional edge between the roots (one of which is the root of B_k).

Prove the following key property of binomial trees: the root of B_k has k children, and the subtree rooted at child $0 \leq i < k$ is B_i (when they are ordered in increasing order of degree). Then show the following simple corollaries:

- B_k has 2^k nodes.
- B_k has height k.

Binomial heaps consist of binomial trees *in heap order* of distinct sizes with additional links: the roots of each tree are in a linked list sorted by degree; and likewise for the nodes at each layer of each binomial tree. (This linked structure also makes it easier to handle children: each node only needs an edge to the leftmost one. For a technical reason, children are ordered in *decreasing* order of degree.)

Prove that a binomial heap with n items contains at most $\log_2(n+1)$ binomial trees. (Proving $O(\log n)$ is also fine.)

The main advantage of binomial over binary heaps is that they support efficient *melding*: given two priority queues, creating a single priority queue with the keys in both.

Give a high-level description of how to implement all main priority queue operations, as well as melding, on a binomial heap in $\Theta(\log n)$ time. Specifically, describe an implementation of `insert()`, `delMax()` or `delMin()` and `meld()`. *Hint: handle melding first.*

C. Algorithm Design

This problem was adapted from the Spring'25 Final Exam.

Given a connected graph G with positive integer edge weights, determine the smallest integer w^* such that removing all edges with weight strictly greater than w^* leaves the graph connected. (Equivalently, w^* is the largest integer such that removing all edged with weight *at least* w^* disconnects the graph.)

The runtime of your algorithm must be $O(E \log E)$, where E is the number of edges in G.