Trees

Drawn with the root at the top and the leaves at the bottom.

Uses

- filesystems on computers (with exceptions like sharing a single file across multiple folders)
- organisational structures (e.g. in UCC)
- mobile phone network backhaul
- phylogenetic trees
- decision trees (e.g. in business)
- computer science algorithms and data structures
- mobile ad hoc network broadcast trees
- propositional logic syntax trees
- analysing structure of natural language sentences

Cycles

A **cycle** is a circuit with the following properties:

- it contains at least one edge
- no edge is visited twice
- no vertex is visited twice except the start/finish vertex

Trees

A **tree** is a connected undirected simple graph with no cycles.

Rooted Trees

A **rooted** tree is a tree in which one of the vertices is designated the **root**, and all edges are then directed away from that root.

Describing vertices in rooted trees

- Parent:
  If there is a directed edge from x to y, then x is the **parent** of y and y is a **child** of x.
- Sibling:
  If two vertices y and w have the same parent, then they are **siblings** of each other.
- Leaf:
  A vertex with no children is a **leaf.** A vertex with children is an

internal vertex.

- Ancestors:
  The **ancestors** of a vertex v are all vertices in the path from v to the root except for v itself.

Properties:

- Adding an edge anywhere to a tree creates a cycle.
- Removing an edge makes it a disconnected graph.
- Every rooted tree has at least one leaf.
- Every tree has (n - 1) edges where n is the number of nodes(?)

Spanning Trees

A **spanning tree** of a graph G=(V,E) is a sub-graph of G that contains every vertex in V.

A simple graph is **connected** if and only if it has a spanning tree.

Minimal Spanning Trees

Consider a simple connected graph with weights on its edges. Different spanning trees will use different edges, and so the costs will be diferent. A Minimal Spanning Tree has the lowest cost.

Informal Algorithm

Pick a vertex. Pick the cheapest edge that links to a vertex we haven't picked yet. Continue until you have the full tree.

Prim's Algorithm

Input: connected undirected graph G = (V, E) with edge weights and n vertices Output: a spanning tree T = (V, F) for G

```
1. T := {v}, where v is any vertex in V
2. F := {}
3. for each i from 2 to n
4.      e := {w, y}, an edge with minimum weight in E such that w is in T
and y is not in T
5.      F := F U {e}
6.      T := [incomplete]
```

Balancing Trees

A tree is more balanced if it's more even from left to right (this is good for guaranteeing search times etc.)