Connected Components
In an undirected graph, the relation $uCv$ iff there is a path from $u$ to $v$ is an equivalence relation (see FCS, p. 467).

- Equivalence classes = connected components.

Why CC's?
Example application: “chips” are built from millions of “rectangles” on 4 or 5 layers of silicon. Certain layers connect electrically.

- Let nodes = rectangles; edges connect electrically connected rectangles.
- CC’s = electrical elements of the chip.
- Deducing electrical elements essential for simulation and other analysis of chip.
  □ Since fabrication and testing is so expensive, computer simulation vital.

Minimum-Weight Spanning Trees

- Attach numerical label to edges.
- Find set of edges of minimum weight (sum of labels) that connect (via a path) every connectable pair of nodes.

Why MWST’s?
Example application: Pricing phone lines.

- By law, a purchase of dedicated lines must be priced proportionally to the weight of the MWST connecting the cities requested.

Representing Connected Components
Data structure = tree with, at each tree node:

1. Parent pointer.
2. Height of the subtree rooted at this node.
• Tree and graph nodes are identified, e.g., use same records or include cross pointers in records for each.

Merge/Find Operations

• \textit{find}(v) finds the root of the tree of which graph node \(v\) is a member.

• \textit{merge}(T_1, T_2) merges trees \(T_1\) and \(T_2\) by making the root of lesser height a child of the other.

CC's Algorithm

1. Start with each graph node in a tree by itself.

2. Look at edges in some order. If edge \(\{u, v\}\) has ends in different trees (use \textit{find} on \(u\) and \(v\) to tell), then \textit{merge} these trees.

• After all edges considered, each tree will be one CC.

Running Time Analysis

Key point: every time a node finds itself on a tree of greater height due to \textit{merge}, the tree also has at least twice as many nodes as its former tree.

• Hence, if there are \(n\) nodes in the graph, paths in trees never get longer than \(\log_2 n\).

• See FCS, p. 472 for proof.

• Consequently, we can consider each of \(m\) edges in \(O(\log n)\) time. Merger, if necessary, takes \(O(1)\). Total = \(O(m \log n)\).

MWST Algorithm

Same as CC's algorithm, but:

• Consider the edges lowest-weight first.

• Proof in FCS, p. 480ff, that the edges resulting in a \textit{merge} form a MWST.

Running Time Analysis

Sort \(m\) edges in \(O(m \log m)\) time.
• Since \( m \leq n^2, \log m \leq 2\log n \), so \( O(m \log n) \) time suffices to sort.

• Thus, MWST’s found in \( O(m \log n) \) time as for CC’s.

Greedy Algorithms
An algorithm that finds a solution by a sequence of steps each of which “seems best at the time” is called greedy.

• Kruskal’s MWST algorithm is “greedy” in this sense.

Class Problem
The Traveling Salesman Problem is to find a simple cycle of minimum weight.

• Does “greedy” work for the TSP?

• How would you implement the greedy approach to TSP? That is, how do you decide whether or not it is OK to add an edge to the selected set?