CS 163 & CS 265: Graph Algorithms

Week 2: Spanning trees

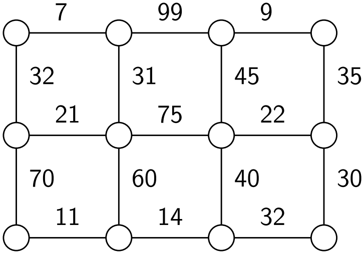
Lecture 2a: Minimum spanning trees

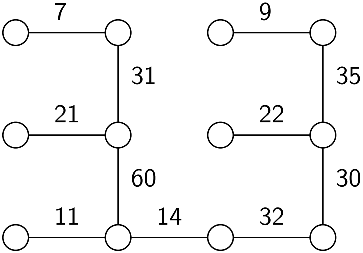
David Eppstein
University of California, Irvine

Winter Quarter, 2024

Definitions

What we want to compute





Input:

- Undirected graph
- Numbers (“weights”) on its edges

Output:

- Tree connecting all vertices
- Minimize total weight

Variations

What if the graph is disconnected?

Result will be a spanning forest, not a tree

It should have the same connected components as input
and form a spanning tree inside each component

Why minimize?

We can also ask for the maximum spanning tree

The same algorithms will work

(or: replace each weight w by its negation $-w$ and minimize)

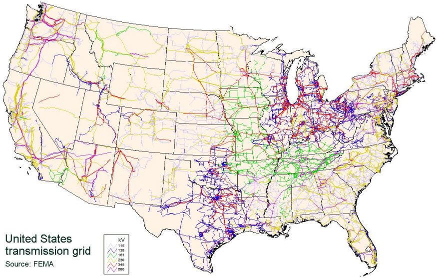
Applications

Original application

Build power grid (network for bulk electricity)

Connecting all the cities and power stations in a country (1920s Czech Republic)

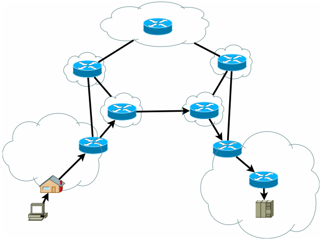
Minimize total construction cost



However, real-world power networks generally have some redundant connections and may have junctions away from cities and stations

Internet routing

Suppose you have a network of computers and routers, with different bandwidths (data per unit time) on different edges



You want data between two machines to follow best possible path

Bandwidth of a path = minimum bandwidth of edges in the path

Path property of maximum spanning trees:
Their paths have the largest possible bandwidth

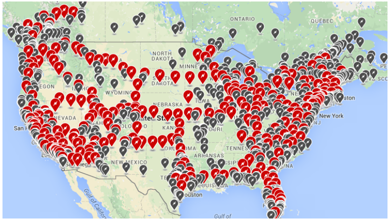
Path property of minimum spanning trees

When we want to avoid paths that use heavy edges

“Bottleneck”: the weight of the heaviest edge on the path

Minimum spanning tree has smallest possible bottleneck

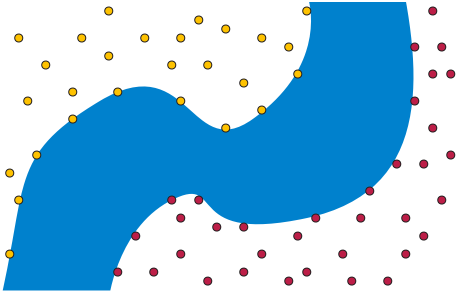
Connect mountain valleys over passes while minimizing the maximum altitude



Drive electric car cross-country avoiding long distances between charging stations

Widest-moat (max intercluster distance) clustering

To divide a set of points into two subsets maximizing the minimum distance between pairs of points from different sets:



Find the minimum spanning tree of a graph connecting all pairs of points, with edge weight = distance

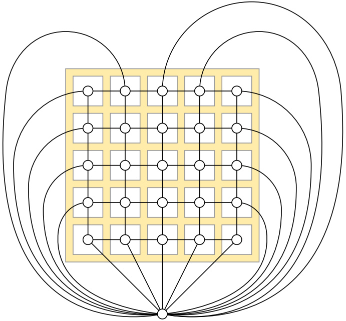
Remove the heaviest edge

Use the resulting two subtrees as clusters

Maze generation

To generate a random maze with only one path between each pair of rooms:

- Make a floorplan of your maze with no doors between any rooms
- Make a graph, vertices = rooms+outside, edges = possible doors, weights = random
- Find its minimum spanning tree and use tree edges as doors



Same model can simulate river networks, flow of water into dry sand

In this context it is called "invasion percolation"

Properties

A simplifying assumption

We will assume that all edge weights are different (no ties)

If true, the minimum spanning tree is unique
(not possible to have two equally good trees)

Important for correctness of some algorithms

When edges might have the same weights:

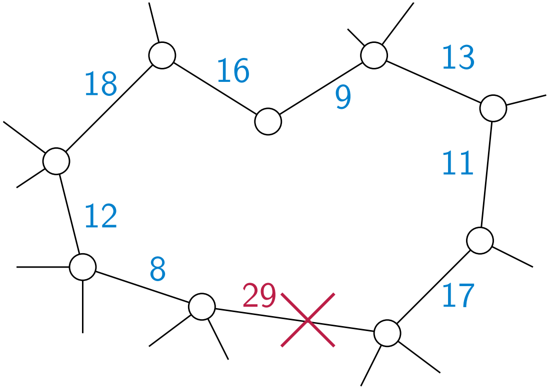
Use a consistent tie-breaking rule

E.g. when edges e and f have the same weight,
order them by their memory address instead

Must define a consistent and unchanging ordering of the edges

Cycle property

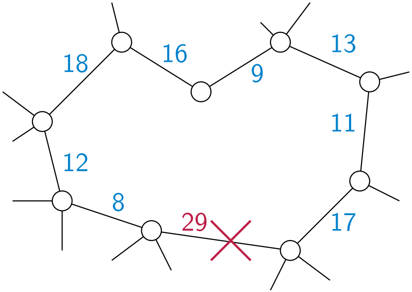
If C is any cycle and e is its heaviest edge
then e cannot be in the MST



We can safely remove it from the graph

Cycle property, restated (easier to prove)

If C is any cycle and e is its heaviest edge
then any tree T that includes e cannot be a MST

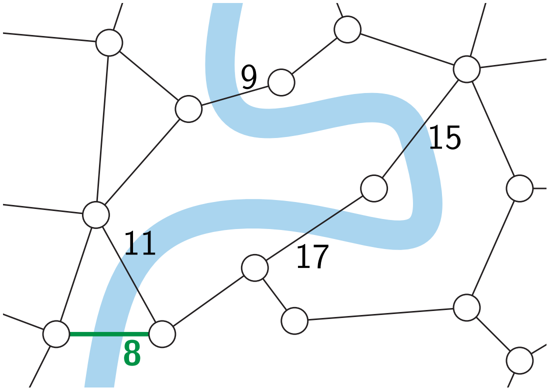


Proof: Remove e from T, breaking it into two subtrees
The rest of C forms a path starting in one subtree, ending in the other,
so at least one edge f connects both subtrees
Add f to the subtrees, making a new tree that is better than T

Cut property

If we cut the vertices of the graph into any two subsets X and $G - X$, and e is the lightest edge with endpoints in both subsets

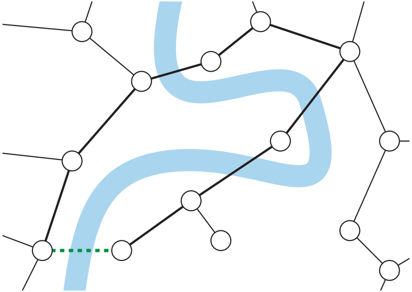
then e must be in in the MST



We can safely add it to the output

Cut property, restated (easier to prove)

If we cut the vertices into any two subsets X and $G - X$, and e is the lightest edge with endpoints in both subsets, then any tree T that avoids e cannot be the MST



Proof: Find the path in T between the endpoints of e

It has one endpoint in X and one in $G - X$

so it must include an edge f that has endpoints in both subsets

$T - f + e$ is a better tree

Proof of the path property

Claim: For any two vertices x and y, the MST path from x to y has the minimum possible weight for its heaviest edge

Proof: Let e be the heaviest edge on the MST path

Remove e from the MST, cutting it into two subtrees X and Y and separating x from y (so $x \in X$ and $y \in Y$)

There cannot be a lighter edge than e from X to Y, because the cut property would force it to be in the tree, but actually e is the only edge from X to Y in the tree

All paths from x to y must cross the same cut somehow, at least once, on an edge at least as heavy as e

Morals of the story

In any weighted connected graph, the minimum spanning tree connects all vertices using a tree with minimum possible weight

The maximum spanning tree is defined and solved in the same way

Both have many applications

Path property: Paths in the minimum spanning tree have the smallest possible bottleneck (weight of heaviest edge in path)

Cycle property: The heaviest edge of a cycle cannot be in the minimum spanning tree

Cut property: The lightest edge from a set X to the rest of the graph $G - X$ must be in the minimum spanning tree