Leaves, trees, forests...

A graph with no cycle is acyclic. An acyclic graph is called a forest.

A connected acyclic graph is a tree.

A leaf (or pendant vertex) is a vertex of degree 1.

A spanning subgraph of G is a subgraph with vertex set $V(G)$.

A spanning tree is a spanning subgraph which is a tree.

Examples. Paths, stars

Properties of trees

Lemma. T is a tree, $n(T) \geq 2 \Rightarrow T$ contains at least two leaves.
Deleting a leaf from a tree produces a tree.

Theorem (Characterization of trees) For an n-vertex graph G, the following are equivalent

1. G is connected and has no cycles.
2. G is connected and has $n-1$ edges.
3. G has $n-1$ edges and no cycles.
4. For each $u, v \in V(G)$, G has exactly one u, v-path.

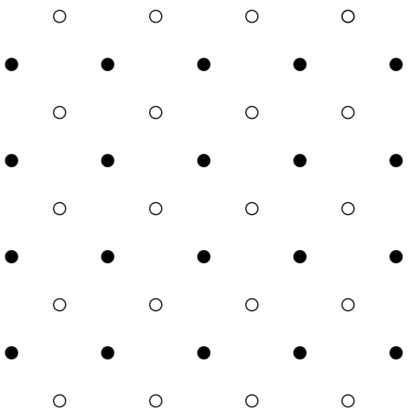
Corollary.

(i) Every edge of a tree is a cut-edge.

(ii) Adding one edge to a tree forms exactly one cycle.

(iii) Every connected graph contains a spanning tree.

Bridg-it* by David Gale



Who wins in Bridg-it?

Theorem. Player 1 has a winning strategy in Bridg-it.

Proof. Strategy Stealing.

Suppose Player 2 has a winning strategy.

Then here is a winning strategy for Player 1:

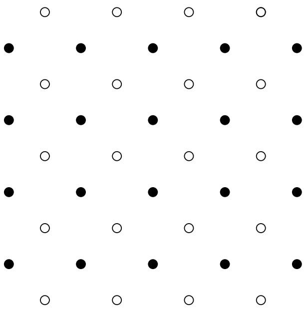
Start with an arbitrary move and then pretend to be Player 2 and play according to Player 2's winning strategy. (Note that playground is symmetric!!) If this strategy calls for the first move of yours, again select an arbitrary edge. Etc...

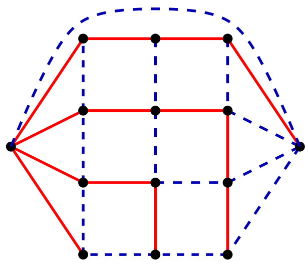
Since you play according to a winning strategy, you win! But we assumed Player 2 also can win $\Rightarrow$ contradiction, since both cannot win.

Good, but HOW ABOUT AN EXPLICIT STRATEGY???*

*In the *divisor-game* strategy-stealing proves the existence of a sure first player win, but NO explicit strategy is known. Similarly for HEX.

An explicit strategy via spanning trees





The game of "Connectivity"

A positional game is played by two players, Maker and Breaker, who alternately take edges of a base graph G. Maker uses a permanent marker, Breaker uses an eraser. Maker wins the positional game "Connectivity" if by the end he occupies a connected subgraph of G. Otherwise Breaker wins.

Theorem. (Lehman, 1964) Suppose Breaker starts the game. If G contains two edge-disjoint spanning tree, then Maker has an explicit winning strategy in "Connectivity".

Proof. Maker maintains two spanning trees T_1 and T_2, such that after each full round,

(i) $E(T_1) \cap E(T_2)$ consists of the edges claimed by Maker,

(ii) $E(T_1) \triangle E(T_2)$ contains only unclaimed edges.

Remark. The other direction of the Theorem is also true.

The tool for Player 1. (i.e. Maker)

Proposition. If T and T' are spanning trees of a connected graph G and $e \in E(T) \setminus E(T')$, then **there is** an edge $e' \in E(T') \setminus E(T)$, such that $T - e + e'$ is a spanning tree of G.

Proposition. If T and T' are spanning trees of a connected graph G and $e \in E(T) \setminus E(T')$, then **there is** an edge $e' \in E(T') \setminus E(T)$, such that $T' + e - e'$ is a spanning tree of G.

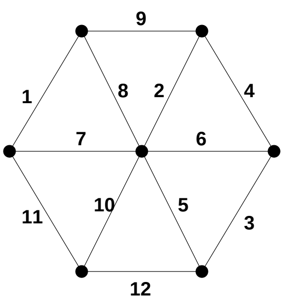
How to build the cheapest road network?

G is a weighted graph if there is a weight function $w : E(G) \rightarrow \mathbb{R}$.

Weight $w(H)$ of a subgraph $H \subseteq G$ is defined as

$$w(H) = \sum_{e \in E(H)} w(e).$$

Example:



Kruskal's Algorithm

Kruskal's Algorithm

Input: connected graph G, weight function $w : E(G) \to \mathbb{R}$, $w(e_1) \leq w(e_2) \leq ... \leq w(e_m)$.

Idea: Maintain a spanning forest H of G. At each iteration try to enlarge H by an edge of smallest weight.

Initialization: $V(H) \leftarrow V(G)$, $E(H) \leftarrow \emptyset$, $i \leftarrow 1$

WHILE $i \leq n$
    $e \leftarrow e_i$
    IF e goes between two components of H THEN
        update $H \leftarrow H + e$
        IF H is connected THEN
            stop and return H
    $i \leftarrow i + 1$

Theorem. In a connected weighted graph G, Kruskal's Algorithm constructs a minimum-weight spanning tree.

Proof of correctness of Kruskal's Algorithm

Proof. T is the graph produced by the Algorithm.
$E(T) = \{f_1, \ldots, f_{n-1}\}$ and $w(f_1) \leq \cdots \leq w(f_{n-1})$.

Easy: T is spanning (already at initialization!)
T is a connected (by termination rule) and has no cycle (by iteration rule) $\Rightarrow$ T is a tree.

But **WHY** is T min-weight?

Let T^* be an arbitrary min-weight spanning tree. Let j be the largest index such that $f_1, \ldots, f_j \in E(T^*)$.

If $j = n - 1$, then $T^* = T$. Done.

Proof of Kruskal, cont'd

If $j < n - 1$, then $f_{j+1} \notin E(T^*)$.
There is an edge $e \in E(T^*)$, such that
$T^{**} = T^* - e + f_{j+1}$ is a spanning tree.

(i) $w(T^*) - w(e) + w(f_{j+1}) = w(T^{**}) \geq w(T^*)$
So $w(f_{j+1}) \geq w(e)$.

(ii) Key: When we selected f_{j+1} into T, e was also available. (The addition of e wouldn't have created a cycle, since $f_1, \ldots, f_j, e \in E(T^*)$.)
So $w(f_{j+1}) \leq w(e)$.

Combining: $w(e) = w(f_{j+1})$, i.e. $w(T^{**}) = w(T^*)$.
Thus T^{**} is min-weight spanning tree and it contains a *longer* initial segment of the edges of T, than T^* did.

Repeating this procedure at most $(n - 1)$-times, we transform any min-weight spanning tree into T.

Some more definitions

The distance between u and v in graph G is

$$d_G(u, v) = \min\{e(P) : P \text{ is a } u,v\text{-path in } G\}.$$

The diameter of G is $diam(G) = \max_{u,v \in V(G)} d(u, v)$.

The eccentricity of a vertex u is $\epsilon(u) = \max_{v \in V(G)} d(u, v)$.

The radius of G is $rad(G) = \min_{u \in V(G)} \epsilon(u)$.