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. G has no loops and has, for each $u, v \in V(G)$, exactly one u,v-path.

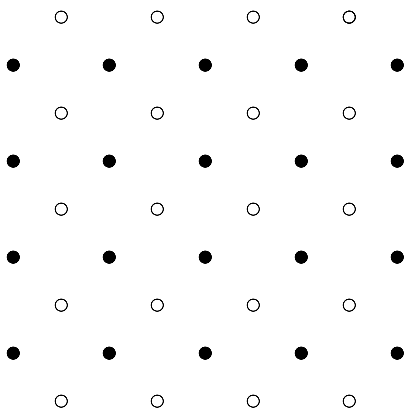
Corollary.

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

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

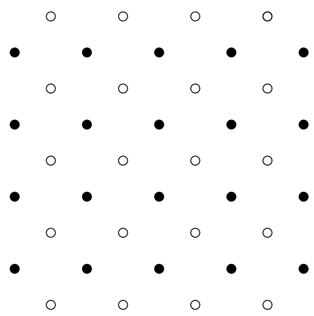
(iii) Every connected graph contains a spanning tree.

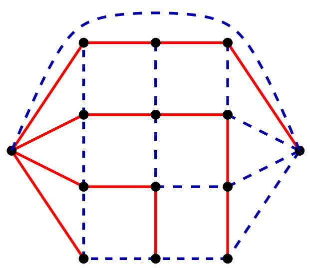
Bridg-it* by David Gale



Bridg-it and spanning trees

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





The tool for Player 1.

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.

Minimum weight spanning tree

Kruskal's Algorithm

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

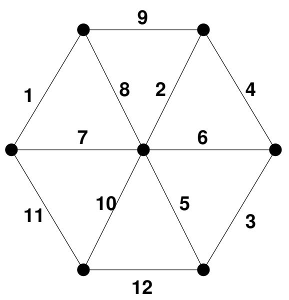
Idea: Maintain a spanning forest H of G. At each iteration try to enlarge H by an edge of smallest weight.

Initialization: $V(H) = V(G)$, $E(H) = \emptyset$

Iteration:
IF e_i goes within one component of H, THEN
    iterate
ELSE
    update $H := H + e$
    IF H is connected THEN
        stop and return H
    ELSE iterate

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

Kruskal's Algorithm — Example



Weighted shortest paths

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 $rad(G) = \min_{u \in V(G)} \epsilon(u)$.

Dijkstra's Algorithm

Input: A (di)graph G, a vertex u, a non-negative weight function $w : \binom{[n]}{2} \to \mathbb{R}$ with $w(xy) = \infty$ if $xy \notin E(G)$.

Maintain:

- a tentative distance $t : V(G) \to \mathbb{R}$ where $t(z)$ is the length of the shortest path yet found.
- a set S to where the shortest path is known and $t(z)$ is the length of the shortest path.

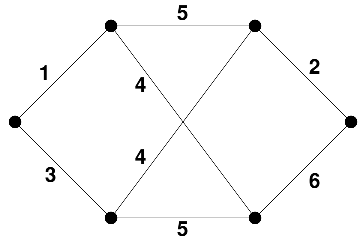
Initialization: $S = \{u\}$, $t(u) = 0$, $t(z) = w(uz)$ for $z \neq u$.

Iteration: Let $v \in V(G) \setminus S$, such that $t(v) = \min_{z \notin S} t(z)$.

Update

- $S := S \cup \{v\}$ and
- $t(z) := \min\{t(z), t(v) + w(vz)\}$ for every $z \in V(G) \setminus S$.

Example and proof of correctness



Theorem. Given a (di)graph G and a vertex $u \in V(G)$, Dijkstra's Algorithm computes $d(u, z)$ for every $z \in V(G)$.

Proof. "Load the induction hypothesis" We prove

1. $t(z) = d(u, z)$ for every $z \in S$ and
2. for every $z \notin S$, $t(z)$ is the smallest length of a u, z-path reaching z directly from S.

Special Case. Breadth First Search (BFS)