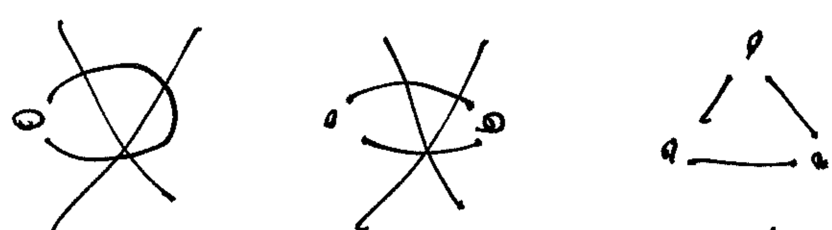
CSE 102 2-25-20

Minimum Weight Spanning tree (MWST)

Defns

- G is <u>connected</u> iff G contains an x-y path (walk also) for all $x, y \in V(G)$
- a <u>cycle</u> is a closed path of length > 0.



smallest cycle

- G is *acyclic* iff it contains no cycles.

- A *tree* is a graph that is both acyclic & connected.

- A spanning tree in G is a subgraph T of G s.t.

  (1) T is a tree, and

  (2) $V(T) = V(G)$.

Suppose G is a weighted graph, i.e. we have

$$w : E(G) \longrightarrow \mathbb{R}$$

<u>Problem</u> (MWST)
given a weighted connected graph, find a sp. tree in G of minimum total weight.

<u>note</u>

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

So T is a MWST iff

$$w(T) \leq w(S)$$

for all sp. trees S in G.

<u>Theorem</u>

G contains a sp. tree iff G is connected.

<u>Proof</u> (exercise.)

<u>notation</u> $G=(V,E)$, $|V|=n$, $|E|=m$.

Prim's Algorithm

- choose an initial vertex (tree.)
- amongst all edges incident with current tree, and whose addition to that tree would not create a cycle, choose one of minimum weight, & add it to current tree
- stop when $n-1$ edges selected.

note:

when tree contains n-1 edges, it includes n vertices, so is a sp. tree.

Theorem

This sp. tree is a MWST.

Proof see book...

Kruskal's algorithm

- choose an edge e of min weight (so $(V, \{e\})$ is a forest.)

- Amongst all edges whose addition to current forest would not create a cycle, choose one of min. weight & add it to current forest.

- stop when $n-1$ edges have been added to forest.

<u>note</u>: by treeness thm,
final spanning forest is connected
hence is a sp. tree.

<u>Theorem</u>

This sp. tree is a MWST
in G.

<u>Proof</u>

let T be the sp. tree
produced by Kruskal, let
S be any other sp. tree in G.

must show: $w(T) \leq w(S)$

let $\{e_1, e_2, \ldots, e_{n-1}\} = E(T)$

in order selected by Kruskal.

so $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_{n-1})$.

If $S = T$, we're done.

Assume that $S \neq T$, so

there is a first edge $e_k \in E(T)$

which is <u>not</u> in S. i.e.

- $\{e_1, e_2, \ldots, e_{k-1}\} \subseteq E(S)$
- $e_k \notin E(S)$

Let $H = S + e_k$ (i.e. $V(H) = V(S) = V(G)$ and $E(H) = E(S) \cup \{e_k\}$.) By trees thm, H contains a unique cycle, call it C.

Note C must contain an edge e of S which is not in T (otherwise C is contained in T, impossible.) Now Remove e from H to obtain

$$R = H - e = S + e_k - e$$

Since R is connected, has $n-1$ edges, n vertices, R is also a sp. tree in G.

<u>claim</u>: $w(e_k) \leq w(e)$ ✓

<u>Proof</u>: e does not form a cycle with $\{e_1, \ldots, e_{k-1}\}$ since $\{e_1, \ldots, e_{k-1}, e\} \subseteq E(S)$. Therefore if $w(e) < w(e_k)$, Kruskal would have chosen e instead of e_k on its k^{th} iteration. $\square$

It follows that

$$w(S) \geq w(R)$$

Also R has (at least) one more edge in common with T than S did.

Repeat this process with R in place of S to obtain a sequence:

$$\underset{\substack{\| \\ R_0}}{w(S)} \geq w(\underset{\substack{\| \\ R_1}}{R}) \geq w(R_2) \geq \ldots \geq w(T)$$

in which each subsequent sp. tree has more edges in common with T, and non-increasing weights.

Eventually we reach T.

$\therefore \quad w(T) \leq w(S)$,

as required. ∎

Matroids & the greedy algorithm

Defn

A <u>matroid</u> is a pair

$$M = (S, \mathbb{I})$$

s.t.

(1) S is finite, non-empty set and $\mathbb{I} \subseteq \mathbb{P}(S)$. The members of $\mathbb{I}$ are called <u>independent sets</u>. (The sets $\mathbb{P}(S) - \mathbb{I}$ are called <u>dependent</u>.)

(2) <u>Hereditary Property</u>:

if $B \in \mathbb{I}$ and $A \subseteq B$, then $A \in \mathbb{I}$.

(3) <u>Exchange Property</u>:

If $A, B \in \mathbb{I}$ and if $|A| < |B|$, then there exists $x \in B - A$ such that

$$A \cup \{x\} \in \mathbb{I}.$$

<u>note</u>: (2) implies $\emptyset \in \mathbb{I}$ (provided $\mathbb{I}$ is itself non-empty.)

<u>Ex.</u> (Matric matroids)

Let P be a rectangular matrix, $S = \{\text{rows of } P\}$, considered as vectors. (If P is $n \times m$, then $v \in S$ is a vector in $\mathbb{R}^m$.)

$$\mathcal{I} = \{A \subseteq S \mid A \text{ is lin. ind. set}\}.$$