Announcements:

Midterm 2: Wed. 10/18 7:00-8:30, Noyes 217
(same time/place as Midterm 1)

Quiz 2: Fri 10/13 (in class)

Optimization: want to minimize or maximize some quantity

Algorithms:

Kruskal's algorithm: find a minimal-weight spanning tree

Dijkstra's algorithm: find a shortest path from u to v

Both are "greedy" algorithms: charge ahead, and don't look back

Let G be a weighted graph w/ nonneg. wts.

If $H \subseteq G$, let $wt(H) = \sum_{e \in E(H)} wt(e)$

The weight of the path/spanning tree/etc. is the sum of the weights of its edges

[Different convention than what we used in the weighted matrix tree thm. related by log]

Kruskal's Algorithm (2.3.1) (for convenience, assume distinct wts)

Input: A weighted conn. graph G

Start: Let $T \subseteq G$ be the subgraph $V(T) = V(G)$, $E(T) = \emptyset$

While T is disconnected:

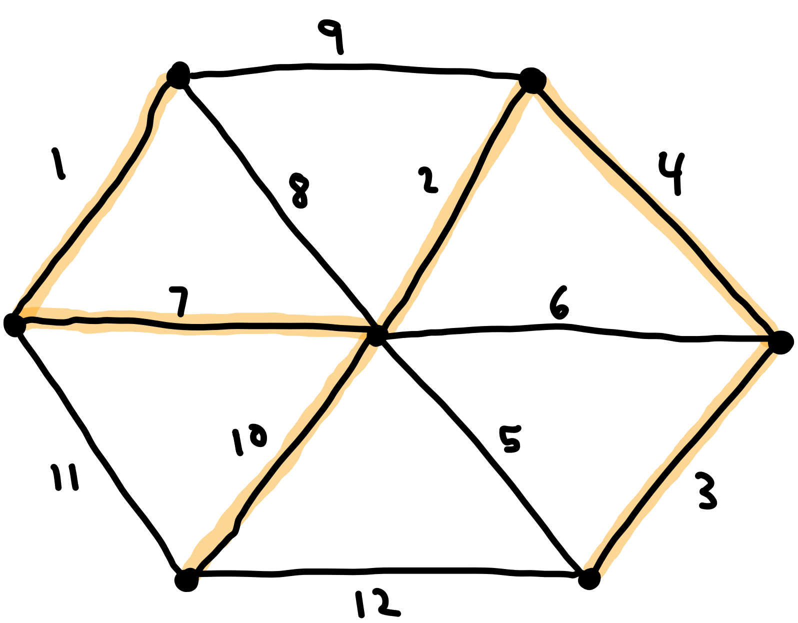
Let e be the least weight of an edge not yet considered

If the endpoints of e are in diff. components of T:
(i.e. if $T \cup e$ is acyclic)

Add e to T

Output: A minimal-weight spanning tree T

Class activity: Kruskal!



$wt(T) = 1 + 2 + 3 + 4 + 7 + 10 = 27$

Thm 2.2.3: The output of Kruskal's Algorithm is always a minimum-weight spanning tree

Pf: Let T be the output of Kruskal's algorithm

First: show T is a tree

Acyclic: If T has a cycle, then let e be the first edge added that created a cycle, and let T' be the graph T before we added e to it. By the algorithm, the endpoints of e are in diff. components of T', which contradicts the assumption that e creates a cycle.

Connected: If T has ≥ 2 conn. components T_1 and T_2, since G is conn. $\exists$ an edge $e \in E(G)$ from T_1 to T_2. But then the endpoints of e lie in diff. components of T, and did so when the algorithm considered e. Thus, $e \in E(T)$, contradicts the assumption that its endpts. lie in diff components of T.

Thus, T is a tree. Let T^* be a min.-wt. spanning tree of G. If $T = T^*$, done; otherwise, let e be the min. wt. edge in $E(T) \setminus E(T^*)$. By Prop. 2.1.7, $\exists e' \in E(T^*) \setminus E(T)$ s.t. $(T^* \cup e) \setminus e'$ is a spanning tree. By minimality of $wt(T^*)$, we must have $wt(e) > wt(e')$. But, this is a contradiction:

Since every edge in T w/ weight $< wt(e')$ is also contained in T^*, and since T^* is acyclic, adding e' to T at the stage of the alg. where we consider it wouldn't have created a cycle, so it should be in T. $\square$

Dijkstra's Algorithm (2.3.5)

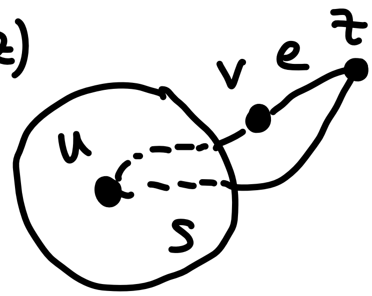
Input: A weighted graph G and a vertex $u \in V(G)$

Start: $S = \{u\}$, $t(u) = 0$,

$$t(z) = \min_{u \overset{e}{\bullet\!\!-\!\!\bullet} z} wt(e) \quad \text{if } z \neq u$$

While $\exists z \notin S,\ t(z) < \infty$:

Choose $v \notin S$ s.t. $t(v) = \min\limits_{z \notin S} t(z)$



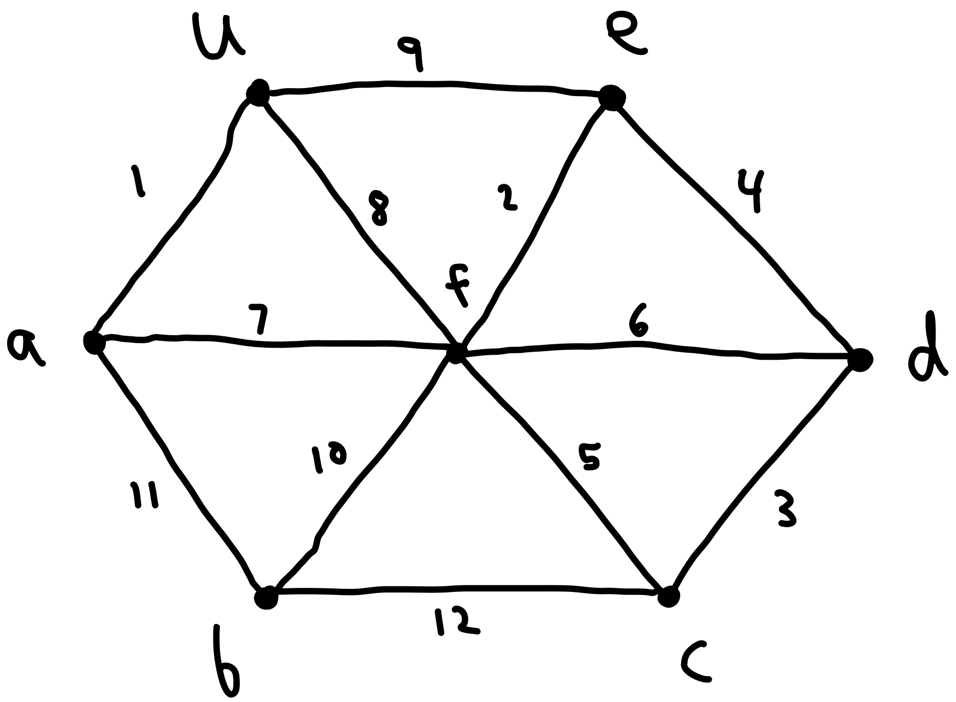
Add v to S

For all edges $v \overset{e}{\bullet\!\!-\!\!\bullet} z$, $z \notin S$:

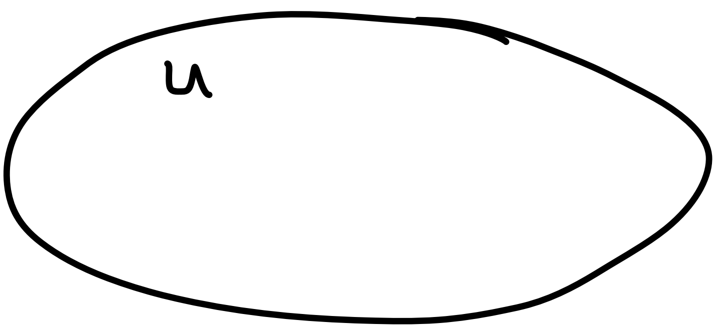
Replace $t(z)$ w/ $\min\big(t(z), t(v) + wt(e)\big)$

Output: $t(v) = d(u,v)$ for all $v \in V(G)$

Class activity: Dijkstra!



S



$t(c) =$

$t(u) =$

$t(d) =$

$t(a) =$

$t(e) =$

$t(b) =$

$t(f) =$

Thm 2.3.7: The output of Dijkstra's Algorithm is always the distance function $d(u,v)$.

Pf: