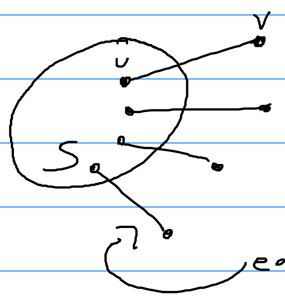
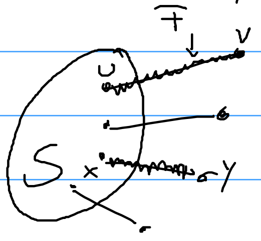
Theorem Consider a connected network (G,w). Let $S\subseteq V(G)$, $F=\{uv\in E(G): u\in S, v\in V(G)\setminus S\}$. Let $uv\in F$. If $w(uv)=\min_{xy\in F} w(xy)$, then there exists a minimum spanning tree of G that contains uv. If $w(uv)<\min_{xy\in F\setminus\{uv\}} w(xy)$, then every minimum spanning tree of G contains uv.



$V(G)\setminus S$

edges in F, edges that go across the "cut" S

Proof. Let T be an arbitrary minimum spanning tree of (G,w). If $uv\in E(T)$ there is nothing to show. If $uv\notin E(T)$, then $E(T)\cup\{uv\}$ contains a cycle and another edge xy in the cycle that is also in F.



Let T' be the spanning tree of G with $V(T')=V(T)$ and $E(T')=(E(T)\setminus\{xy\})\cup\{uv\}$.
If $w(uv)=\min_{st\in F} w(st)$, then the weight of T' is no larger than that of T, so T' is a minimum spanning tree.
If $w(uv)<\min_{st\in F\setminus\{uv\}} w(st)$, then T' has strictly smaller weight than T, a contradiction to the assumption that T is a minimum spanning tree. Therefore every minimum spanning tree of (G,w) contains uv. $\square$

Theorem When applied to a connected network, Prim's algorithm finds a minimum spanning tree.

Proof. Any edge e added by the algorithm has minimum weight in $F = \{uv \in E : u \in V(T), v \in V(G) \setminus V(T)\}$, which is equal to F in the previous theorem for $S = V(T)$.
If e is the unique edge of minimum weight in F, then by the previous theorem it is contained in every minimum spanning tree and can safely be taken.
To see that it is safe to take e even if it is not the unique edge with minimum weight in F, imagine that we add a small value $\varepsilon > 0$ to the weight of every edge of minimum weight in $F \setminus \{e\}$. As we only increased the weight of edges that the algorithm has not yet taken, it would have made the same choices for the original and the modified network up to the point where it considers e. Since e is now the unique edge of minimum weight in F, by the previous theorem, it is safe to take, and we will obtain a minimum spanning tree of the modified network that contains e.

original network	modified network

number of vertices

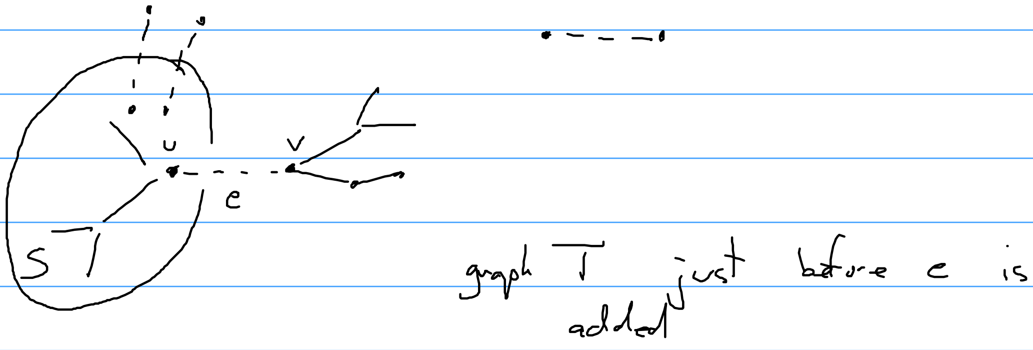
$$\begin{array}{l|l|l}
\text{weights of minimum spanning trees} & w_1 & \leq w_1 + n \cdot \varepsilon \\
 & \shortparallel & \\
 & w_2 & \leq w_2 + n \cdot \varepsilon \\
 & \shortparallel & \\
 & w_3 & \leq w_3 + n \cdot \varepsilon \\
 & \wedge & \wedge \leftarrow \text{for some } \varepsilon > 0 \\
\text{weights of spanning trees that are not minimum spanning trees} & w_4 & \geq w_4 \\
 & \vdots & \vdots \\
 & w_e & \geq w_e
\end{array}$$

If $\varepsilon > 0$ is chosen small enough, then any spanning tree that was not a minimum spanning tree of the original network is not a minimum spanning tree of the modified network. The spanning tree that we made the algorithm choose, and which contains e, will therefore be a minimum spanning tree of the original network.

$\square$

Theorem When applied to a connected network, Kruskal's algorithm finds a minimum spanning tree.

Proof sketch



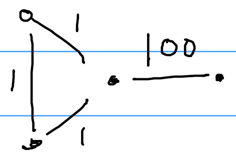
make S the set of vertices in the connected component of T that contains u

Claim: e has minimum weight among edges with exactly one endpoint in S

Then use ε trick.

$\square$

Theorem Consider a connected network (G, w). Let $F \subseteq E(G)$ be the set of edges contained in a cycle of G. Let $uv \in F$. If $w(uv) = \max_{xy \in F} w(xy)$, then there exists a minimum spanning tree of G that does not contain uv. If $w(uv) > \max_{xy \in F \setminus \{uv\}} w(xy)$, then no minimum spanning tree of G contains uv.



Proof. Let T be a minimum spanning tree of (G,w). If $e \notin E(T)$, there is nothing to show. If $uv \in E(T)$, then there exists another edge $xy \in F$ such that $xy \notin E(T)$. Let T' be the spanning tree with $V(T') = V(T)$ and $E(T') = (E(T) \setminus \{uv\}) \cup \{xy\}$. If $w(uv) = \max_{xy \in F} w(xy)$, the weight of T' is no larger than the weight of T, so T' is a minimum spanning tree, and it does not contain uv.

If $w(uv) > \max_{xy \in F \setminus \{uv\}} w(xy)$, then T' has strictly smaller weight than T, a contradiction to the assumption that T is a minimum spanning tree. This means there exists no minimum spanning tree that contains uv. □

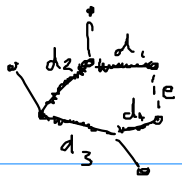
Theorem When applied to a connected network, the reverse-delete algorithm finds a minimum spanning tree.

Proof sketch
Claim: Every edge removed by the algorithm has maximum weight on a cycle.
Then use ε trick.

Theorem Let (G,w) be a network, T a minimum spanning tree of (G,w). Then T is the unique minimum spanning tree of (G,w) if and only if the following holds: for every $e \in E(G) \setminus E(T)$ and every edge $d \in E(T)$ on the unique u-v-path in T,

$$\underset{\text{not in } T}{\underset{\uparrow}{w(uv)}} > \underset{\text{in } T}{\underset{\uparrow}{w(d)}}$$

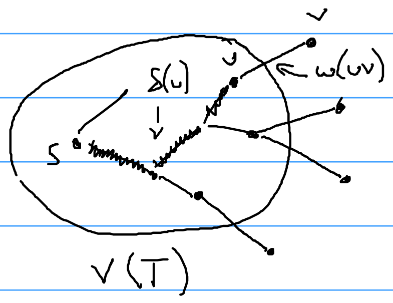
Proof in exercises

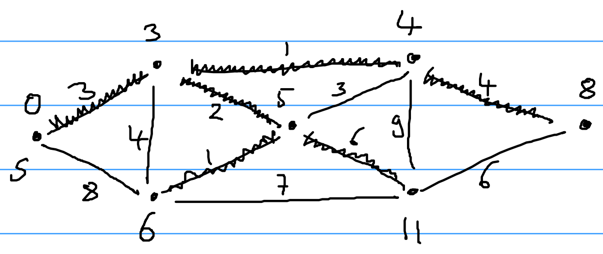


5.2 Shortest Paths for Non-negative Weights

Algorithm (Dijkstra) Consider a network (G, w) and $s \in V(G)$. Dijkstra's algorithm starts from the tree T with $V(T) = \{s\}$ and $E(T) = \emptyset$ and then repeat the following steps:

1. Let $F = \{uv \in E(G) : u \in V(T),\ v \in V(G) \setminus V(T)\}$
2. For each $u \in V(T)$, let $\delta(u)$ be the length of the unique s-u-path in T.
3. Let $uv \in F$ such that
$$[\delta(u) + w(uv)] = \min_{xy \in F} [\delta(x) + w(xy)]$$
4. Add v to $V(T)$ and uv to $E(T)$.





Running time:
- at most $|V(G)| - 1$ iterations
- store $\delta(u)$ for each $u \in V(T)$
- uv in Step 3 can be found in $O(|E(G)|)$ basic operations

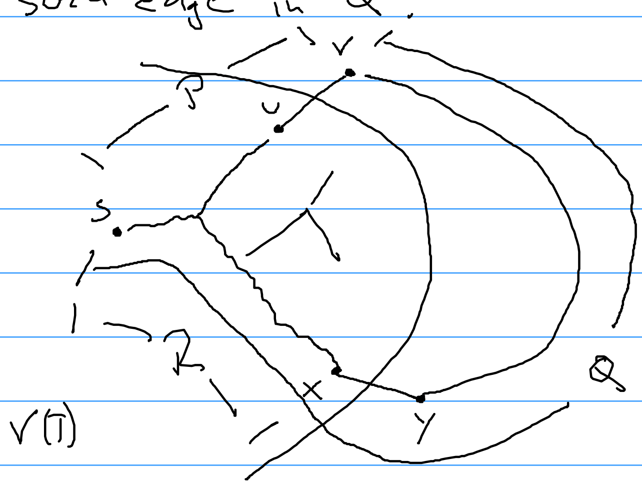
Overall running time $O(|V(G)| \cdot |E(G)|)$

<u>Theorem</u> When applied to a network (G,w) with non-negative weights starting from $s \in V(G)$, Dijkstra's algorithm produces a spanning tree of the connected component of G containing s such that for all vertices u in the tree, the unique s-u-path in the tree is a shortest s-u-path in (G,w).

Proof. We show the claim by induction on $|V(T)|$: throughout the algorithm, T contains a shortest s-t-path for every $t \in V(T)$. Initially $V(T)=\{s\}$, and the claim holds trivially.

Now assume that the algorithm has constructed a tree T that contains a shortest s-t-path for every $t \in V(T)$ and selects uv for addition to the tree.

Let P be the s-v-path that follows the unique s-u-path in T and then traverses the edge uv. For any path Q, let $\ell(Q)$ be the length of Q. It suffices to show that P is a shortest s-v-path in (G,w). Assume for contradiction that there exists an s-v-path Q with $\ell(Q) < \ell(P)$. Since $s \in V(T)$ and $v \in V(G) \setminus V(T)$, Q must contain an edge in F. Let xy be the first such edge in Q.



Let R be the s-x-path obtained by following Q to x.

Because weights are non-negative,

$$\ell(Q) \geq \ell(R) + w(xy)$$

Since the algorithm chooses $uv \in F$ over $xy \in F$,

$$\delta(x) + w(xy) \geq \delta(u) + w(uv)$$

Since $u, x \in V(T)$, and by the induction hypothesis

length of some s-x-path $\rightarrow$ $\ell(R) \geq \delta(x)$ $\leftarrow$ length of shortest s-x-path

and $\ell(P) = \delta(u) + w(uv)$

Therefore
$$\begin{aligned} \ell(Q) &\geq \ell(R) + w(xy) \\ &\geq \delta(x) + w(xy) \\ &\geq \delta(u) + w(uv) \\ &= \ell(P) \end{aligned}$$
, a contradiction. $\square$