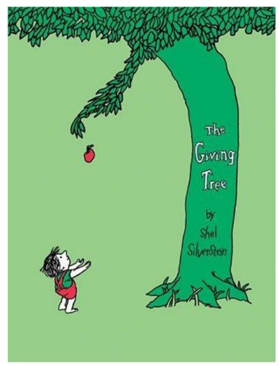
Ma/CS 6a

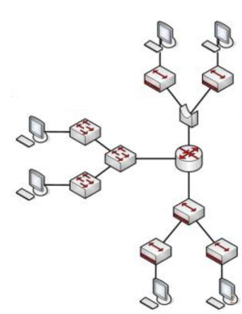
Class 10: Spanning Trees



By Adam Sheffer

Problem: Designing a Network

- **Problem.** We wish to rebuild Caltech's communication network.
  - We have a list off all the routers, and the cost of connecting every pair of routers (some connections might be impossible).
  - We wish to obtain a connected network, while minimizing the total cost.

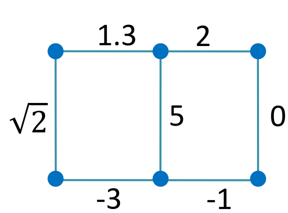


A Solution?

- We wish to build a graph $G = (V, E)$. What are the vertices of V?
  - A vertex for every router.
- What are the edges of E?
  - An edge between every two routers that can be connected.
- Is the graph directed?
  - No.
- Where are the connection prices presented in the graph?

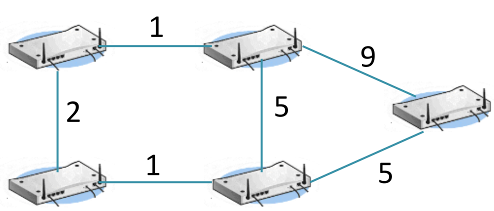
Weighted Graphs

- Given a graph $G = (V, E)$, we can define a ***weight function*** over the edges
$$w: E \to \mathbb{R}.$$
  - That is, for every edge $e \in E$, the ***weight*** of e is denote as $w(e)$.



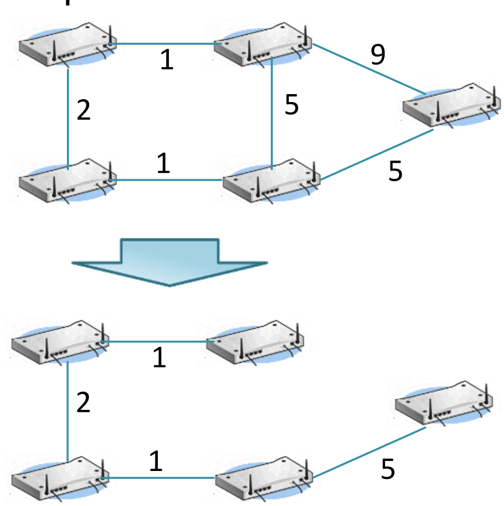
Back to Desiging a Network

- We have an undirected graph $G = (V, E)$ whose vertices represent the routers and edges represent possible connections.
- How are the costs related to the graph?
  - The weight of every edge should be the corresponding cost.



Example: Network Design

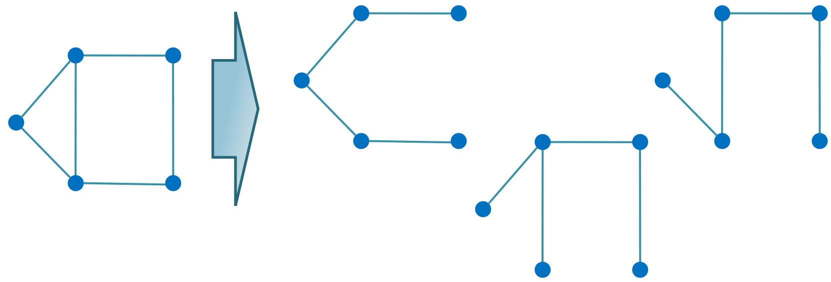
- What's the cheapest solution in this case?



Network cost:9

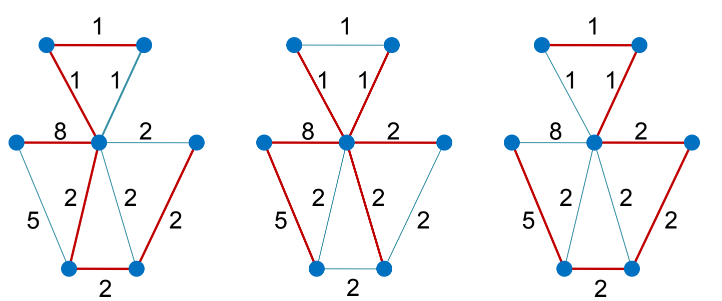
Spanning Trees

- A *spanning tree* is a tree that contains all of the vertices of the graph.
- A graph can contain many distinct spanning trees.



The Weight of a Spanning Tree

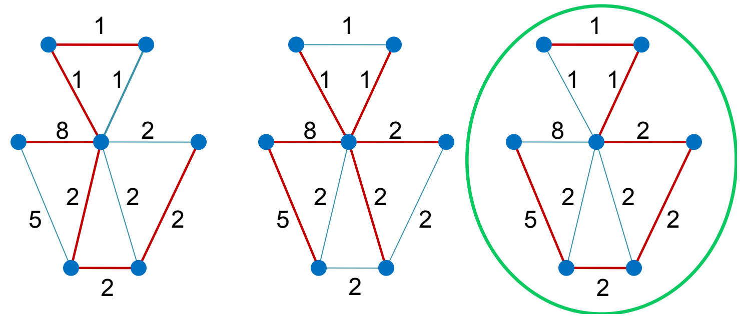
- The weight $w(T)$ of a spanning tree T is the sum of its edge weights.



$w(T) = 16$ $w(T) = 19$ $w(T) = 13$

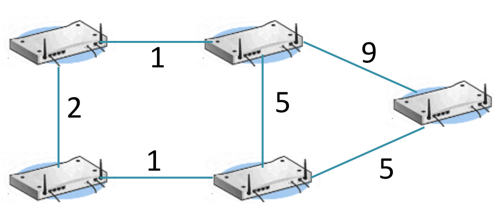
Minimum Spanning Tree

- Given a connected undirected graph $G = (V, E)$ and a weight function $w: E \to \mathbb{R}$, a ***minimum spanning tree*** (or, *MST*) is a spanning tree of G of a minimum weight.

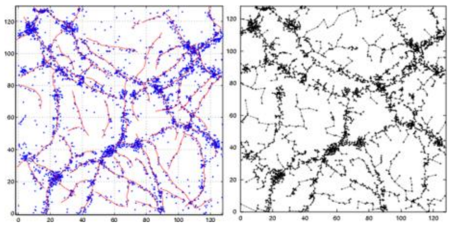


Routers and MSTs

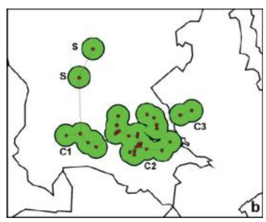
- In the network design problem we are looking for an MST of the graph.



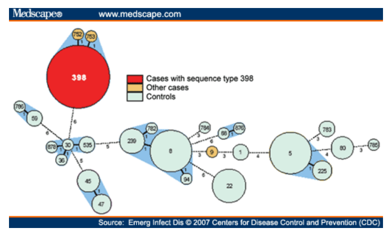
MSTs Are Also Useful For…



Galaxies research



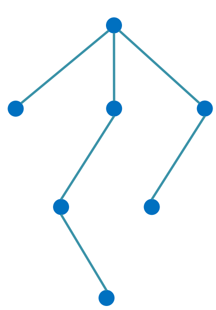
Cacti species in the desert

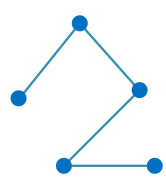


Genetic research

The Size of a Spanning Tree

- Given a graph $G = (V, E)$, how many edges are in each of its spanning trees?
  - Exactly $|V| - 1$.



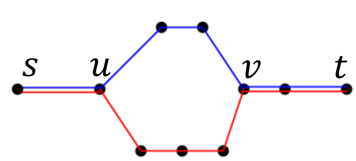


Graphs with $|V| - 1$ Edges

- **Claim.** Consider a graph $G = (V, E)$ containing no cycles and with $|E| = |V| - 1$. Then G is a spanning tree.
- **Proof.** By induction on $|V|$.
  - Induction basis. Obvious when $|V| = 1$.
  - Induction step. Since there are no cycles in G, there must be a vertex $v \in V$ of degree 1.
  - Remove v and the edge e adjacent to it. By the induction hypothesis, the resulting graph is a spanning tree. After reconnecting v and e, we still have a spanning tree.

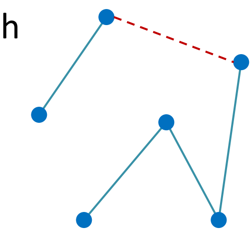
Spanning Trees: Unique Paths

- **Claim.** In any spanning tree T there is *exactly* one path between any two vertices.
  - Assume, for contradiction, that there are two paths P, Q in T between vertices s and t.
  - u – the last common vertex before the paths P, Q split (when traveling from s to t).
  - v – the first vertex common to both paths after u.
  - The portions of P and Q between u and v form a cycle. *Contradiction!*



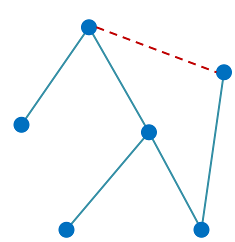
Spanning Trees: Removing an Edge

- **Claim.** Removing any edge of a spanning tree splits it into a forest of two trees.
- **Proof.** $e = (u, v)$ – the removed edge.
  - The edge e was the unique path between u and v. After e's removal there is no path between u and v. Thus, at least two trees.
  - Every vertex whose unique path to u uses (u, v) remains connected to v. Every other vertex remains connected to u. Thus, exactly two trees.



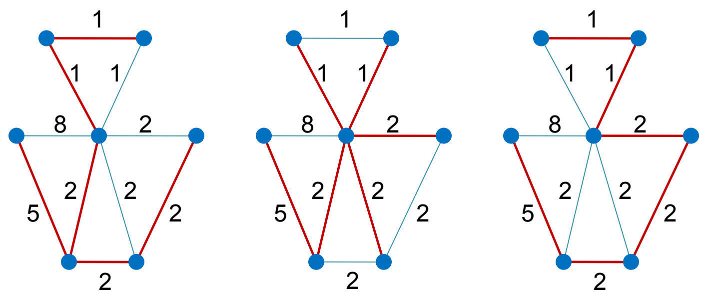
Spanning Trees: Adding an Edge

- **Claim.** Adding an edge to a spanning tree yields ***exactly*** one cycle.
- **Proof.** Homework exercise.

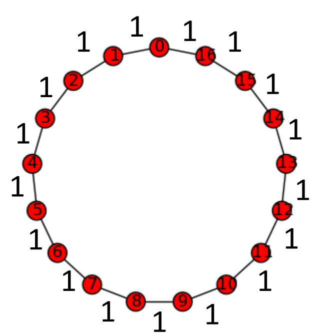


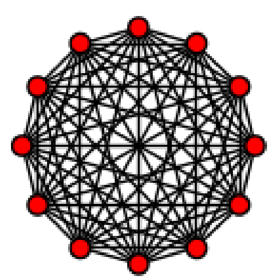
How Many MSTs Are There?

- What is the maximum number of MSTs that a graph can contain?



A Lot of MSTs





$|V|$ *MSTs*

$|V|^{|V|-2}$ *MSTs*

(try to prove: at least $(|V|-1)!$)

(Not so) Comic Relief

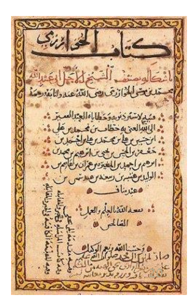
- What is the origin of the word *algorithm*?

(Not so) Comic Relief

- What other major mathematical concept has a similar origin?

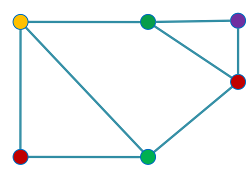
- Al Khwarizmi's book is called

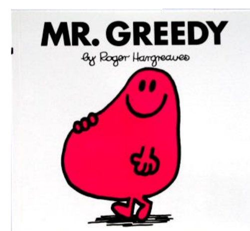
Al-Kitāb al-mukhtaṣar fī hīsāb al-ğabr wa'l-muqābala



Greedy Algorithms

- A *greedy algorithm* makes the locally optimal choice at each stage, without having a long-term strategy.
  - Might not yield the optimal result.
  - **Example.** The coloring algorithm for graphs of maximum degree k.

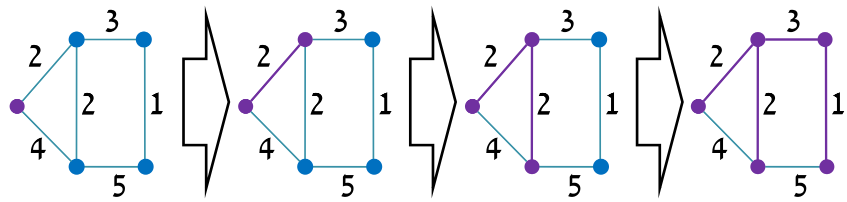




Prim's MST Algorithm

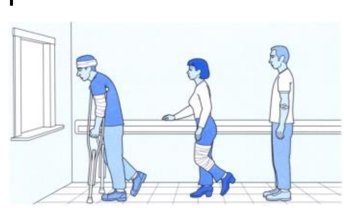
- Given a connected graph $G = (V, E)$, we find an MST by using a greedy approach. We gradually grow a tree T until it becomes an MST.
- Choose an arbitrary vertex $r \in V$ to be the root and set $T \leftarrow r$.
- As long as T is not a spanning tree:
  - Find the lightest edge e that connects a vertex of T to a vertex v not in T. Add v and e to T.

Example: Prim's Algorithm



Data Structure: Priority Queue

- A *priority queue* stores "objects" (in our case – vertices). Each object has a priority.
- Supports the operations:
  - Enqueue – insert an object to the the queue.
  - Dequeue – remove the object with the *highest priority* from the queue.



Prim's Algorithm in More Detail

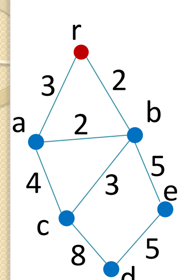
$Q \leftarrow$ an empty priority queue
foreach vertex $v \in V \setminus \{r\}$ **do**
  $key[v] \leftarrow \infty$
$key[r] \leftarrow 0$
$\pi[r] \leftarrow \mathrm{NIL}$
insert V to Q using key as priorities
while Q is not empty **do**
  $u \leftarrow \texttt{Extract-Min}(Q)$
  foreach neighbor v of u **do**
    if $v \in Q$ and $w(u, v) < key[v]$ **then**
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$
$A \leftarrow \{(\pi[u], u) \mid u \in V \setminus \{r\}\}$
return (V, A)

$key[v]$ is the best known weight for connecting v to the tree.

As in BFS, $\pi[v]$ is the parent of v in the tree.

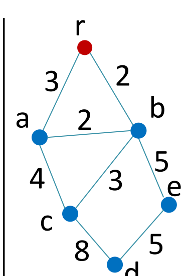
Q- priority queue according to the values of $key[v]$.

Another Prim Run

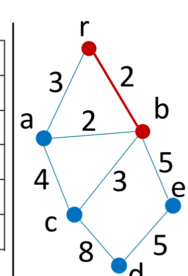


Q

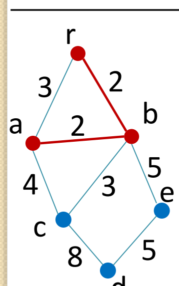
r	0
a	∞
b	∞
c	∞
d	∞
e	∞



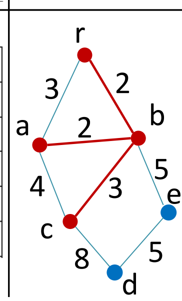
b	2
a	3
c	∞
d	∞
e	∞



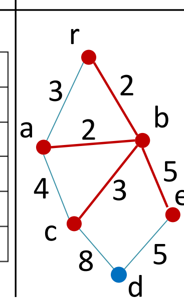
a	2
c	3
e	5
d	∞



c	3
e	5
d	∞



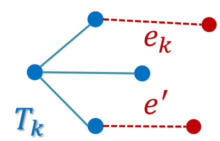
e	5
d	8



d	5

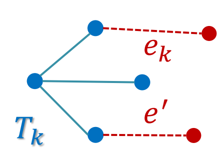
Correctness of Prim

- Let T be a spanning tree that Prim returns. Assume, for contradiction, that there exists an MST M such that $w(T) > w(M)$.
- Let $e_1, \dots, e_n$ be the edges of T in their insertion order.
- Let e_k be the first edge of T that is not in M.
- T_k – the tree consisting of $e_1, e_2, \dots, e_{k-1}$.
- We claim that there exists an edge $e' \in M$ such that $w(e_k) \leq w(e')$ and that e' connects a vertex of T_k with a vertex not in T_k.



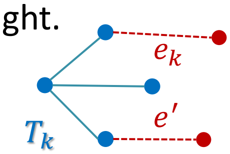
Proving the Claim

- T_k – a tree consisting of $e_1, e_2, \dots, e_{k-1}$.
- **Claim.** There exists an edge $e' \in M$ such that $w(e_k) \leq w(e')$ and e' connects a vertex of T_k with a vertex not in T_k.
  - Write $e_k = (u, v)$. Let P be the path in M between u and v.
  - Let e' be an edge of P that connects a vertex of T_k with a vertex not in T_k.
  - We have $w(e_k) \leq w(e')$, since otherwise Prim would have chosen e' before e_k.



Correctness of Prim (cont.)

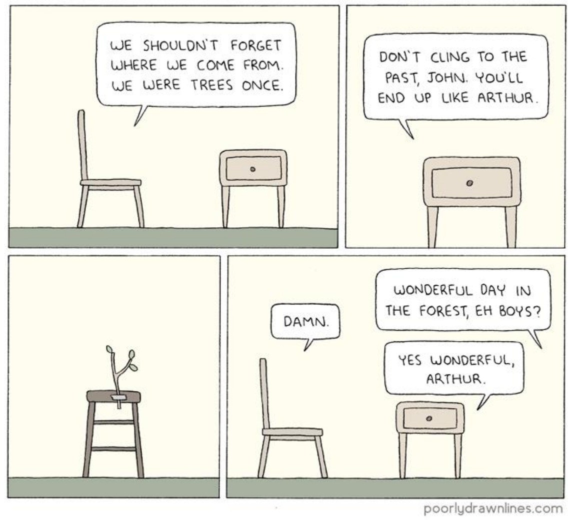
- Let T be a tree that Prim returns.
- Assume, for contradiction, that there exists an MST M such that $w(T) > w(M)$.
- Let e_k be the first edge of T that is not in M.
- There exists $e' \in M$ that connects a vertex of T_k with a vertex not in T_k and $w(e_k) \leq w(e')$
- Adding e_k to M creates a cycle that also contains e'. Removing e' results in a spanning tree M' of a smaller or equal weight.
- Thus, M' is another MST.



Correctness of Prim (end)

- Let T be the tree that Prim returns. Let $e_1, \ldots, e_n$ be the edges of T in their insertion order.
- We started with an MST that contains the edges $e_1, \ldots, e_{k-1}$ but not e_k and found an MST that contains the edges $e_1, \ldots, e_{k-1}, e_k$.
- Repeating this process, we eventually obtain an MST that contains $e_1, \ldots, e_{|V|-1}$.
- That is, T is an MST!

The End: A common Syndrome of Spanning Trees



Can anyone make me a variant of this with spanning trees? ☺