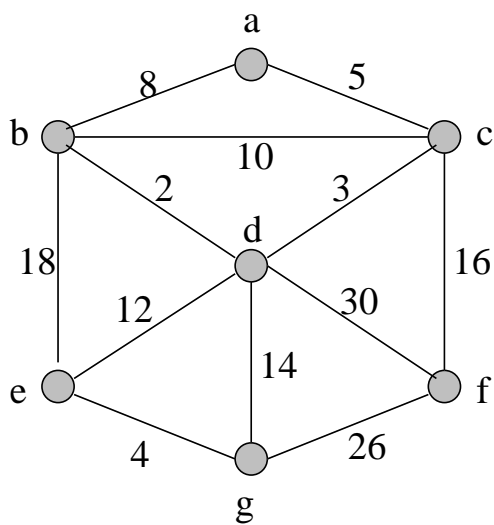


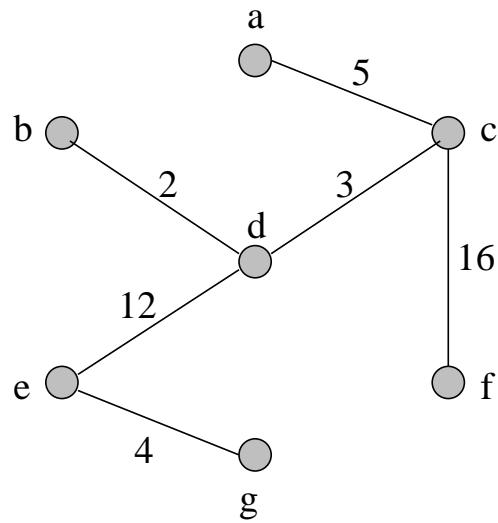
# Minimum Spanning Trees

---

- Given an *undirected* graph  $G = (V, E)$ , with edge costs  $c_{ij}$ .
- A spanning tree  $T$  of  $G$  is a cycle-free subgraph that spans all the nodes.
- The *cost* of  $T$  is the *sum* of the costs of the edges in  $T$ .
- MST is the smallest cost spanning tree.



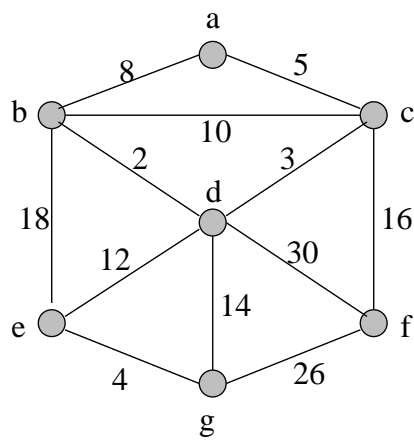
*Graph G*



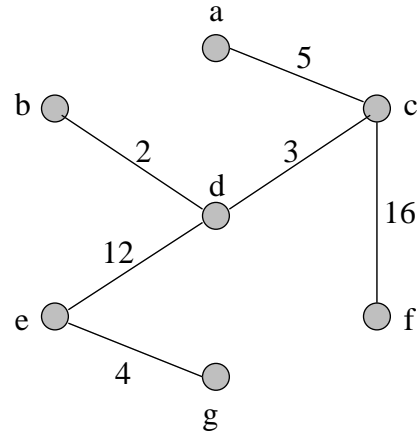
*MST*

# Applications of MST

---



*Graph G*



*MST*

- **Direct applications: interconnection of entities.**
  1. electrical devices (circuit boards)
  2. utilities (gas, oil)
  3. computers or communication devices by high speed lines.
  4. cable service customers
- **Indirect applications.**
  1. Optimal message passing.
  2. Data storage.
  3. Cluster analysis

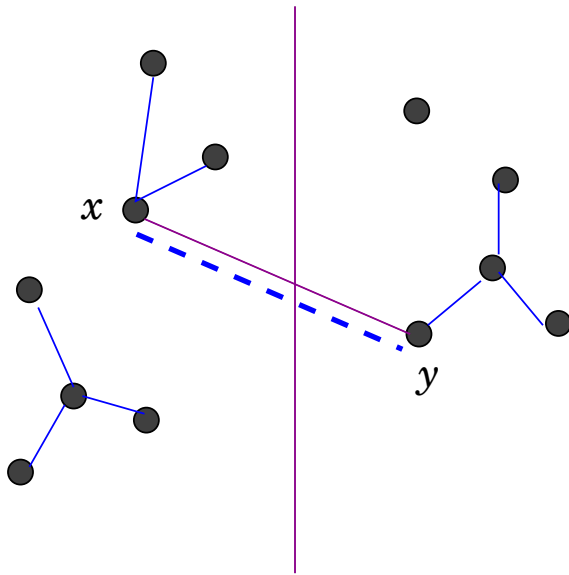
# Optimality Conditions

---

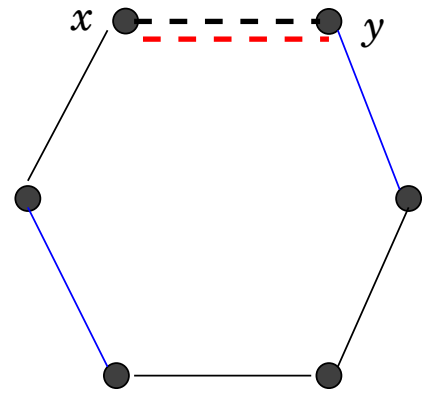
- Greedy incremental flavor: add one edge at a time.
- Each step colors an edge of  $G$  blue (accept) or red (reject).
- **Color Invariant (CI):**  $\exists$  MST containing all blue edges, and no red edges.
- Recall that a cut in  $G = (V, E)$  is a partition of its vertices  $(X, V - X)$ .

# Blue and Red Rules

---



*Cut Rule (Blue)*



*Red Rule (Cycle)*

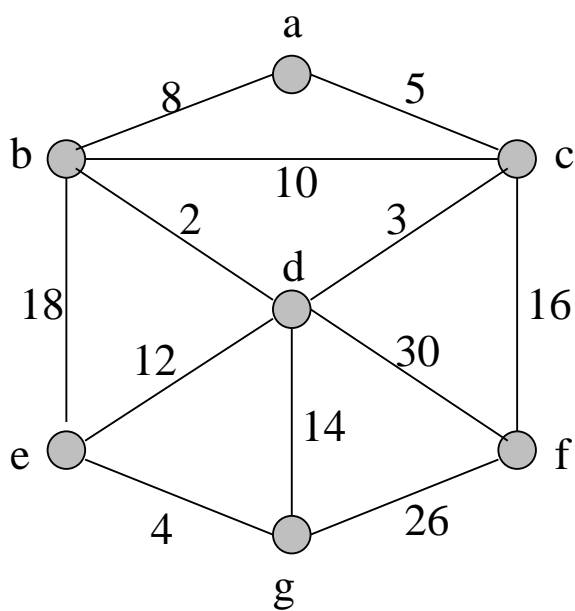
**Blue (Cut) Rule:** Select a cut not crossed by any blue edge. Among the uncolored edges crossing the cut, make the minimum cost edge blue.

**Red (Cycle) Rule:** Select a simple cycle with no red edges. Among all uncolored edges of the cycle, make the maximum cost one red.

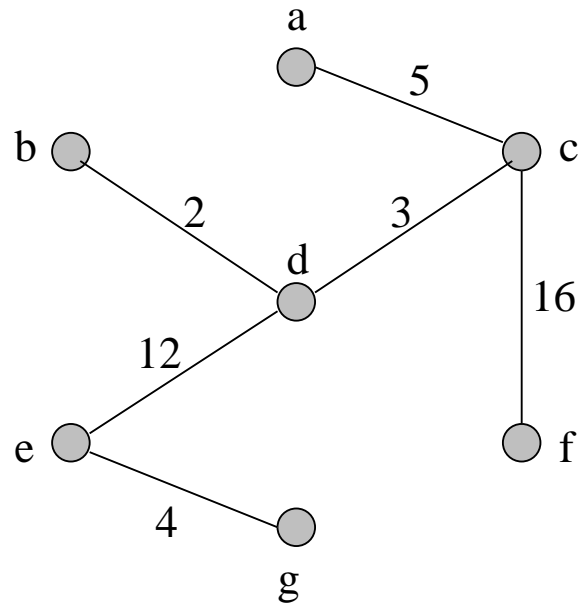
# Generic MST Algorithm

---

**Theorem:** *Apply red and blue rules in arbitrary order until neither rule applies. The resulting set of blue edges forms a MST.*



*Graph G*



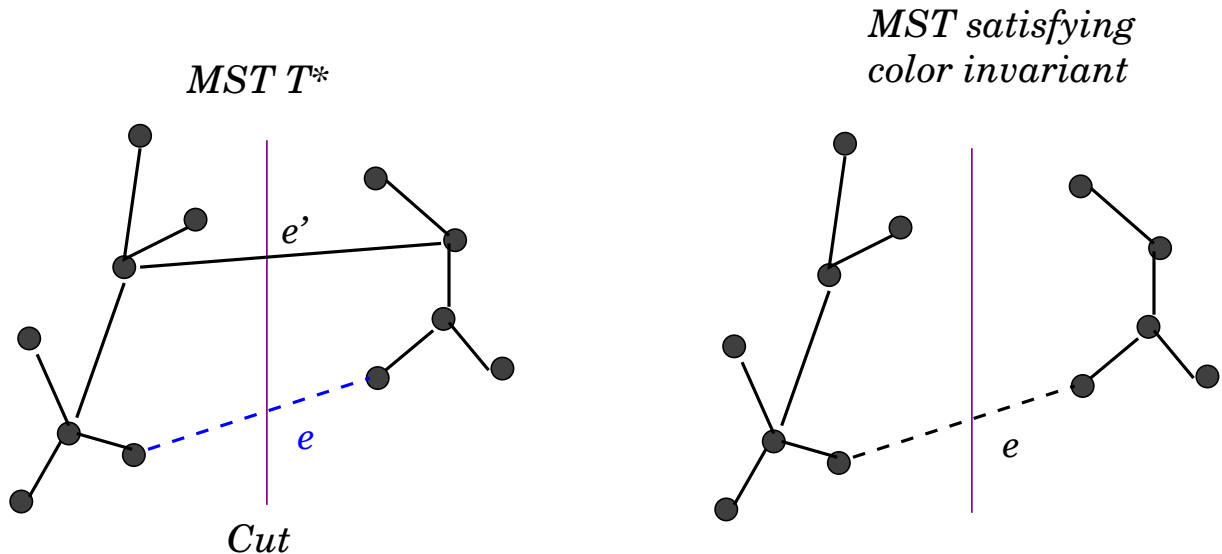
*MST*

- Proof has two parts:
  1. The Color Invariant (CI) holds.
  2. All edges are ultimately colored.

# Correctness of Blue Rule

---

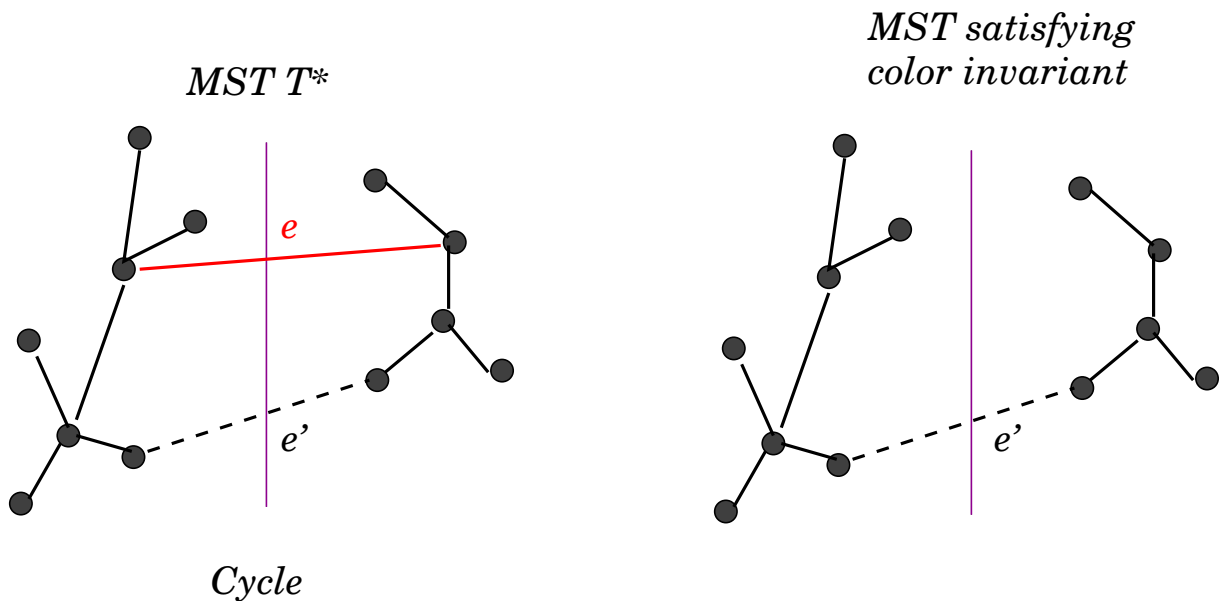
- Let  $T^*$  be the MST guaranteed by the CI before the last coloring step.



- Suppose the last step was to color  $e$  blue. Consider the cut  $(X, V - X)$  to which blue rule applied. Some edge  $e'$  of  $T^*$  must cross this cut.
- The graph  $T^* + e$  contains a cycle containing both  $e$  and  $e'$ , and  $cost(e) \leq cost(e')$ . (Why?)
- So  $T^* + e - e'$  is also a MST.

# Correctness of Red Rule

---



- Suppose edge  $e$  colored red. If  $e \notin T^*$ , then  $T^*$  still satisfies CI. Otherwise, consider  $T^* - e$ . It has two components.
- The cycle used in coloring  $e$  has some edge  $e'$  with one end in each of these components.
- By choice,  $cost(e') \leq cost(e)$ . Thus, the tree  $T^* - e + e'$  is also an MST.

# Proof of Completion

---

- Blue edges form a forest.
- Suppose edge  $e$  left uncolored, and neither rule applies.
- If both endpoints of  $e$  in same blue tree  $B$ , then red rule applies to the cycle  $B + e$ .
- If endpoints in different blue trees  $B_1, B_2$ , then blue rule applies across the cut separating  $B_1$  and  $B_2$ .
- Thus, the generic MST algorithm is correct.

# Prim's Algorithm

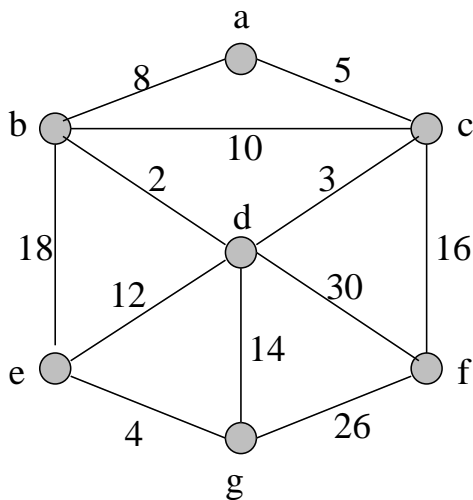
---

- Start at any node  $s$ , and set  $T = \{s\}$ .
- repeat  $n - 1$  times

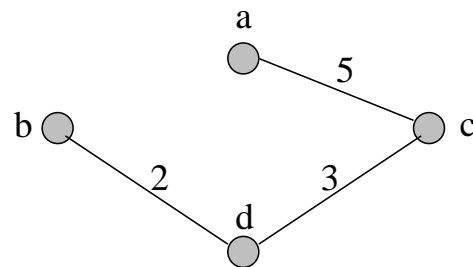
Let  $T$  be the current tree.

Choose a minimum cost *uncolored*  $(u, v)$  with  $u \in T$  and  $v \notin T$ .

Color  $(u, v)$  **blue**, and add  $v$  to  $T$ .



Graph G



After 3 steps.

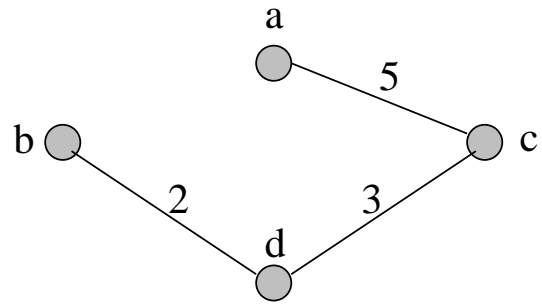
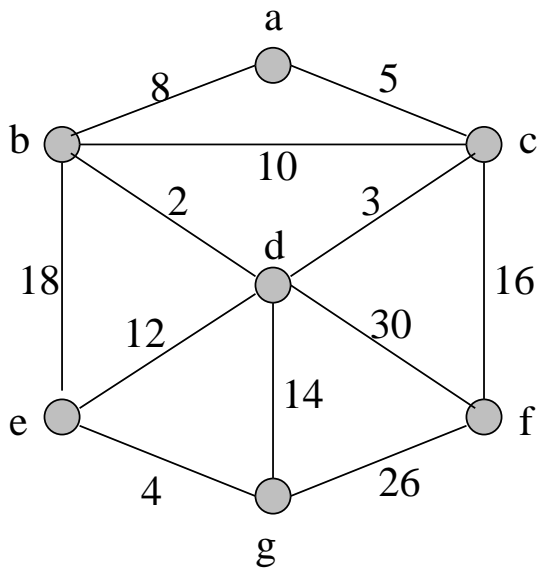
# Prim's Algorithm

---

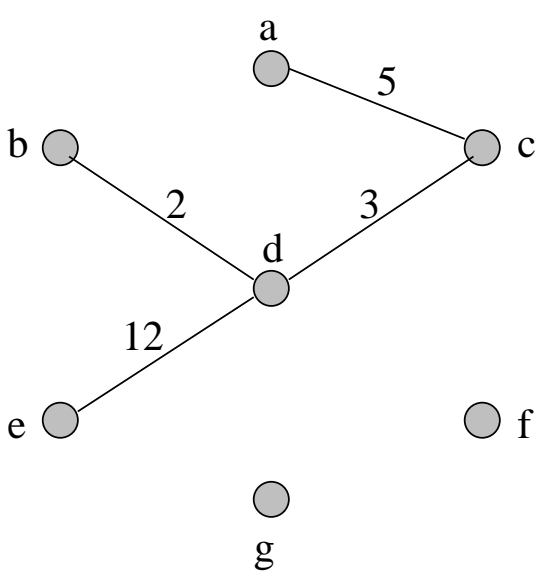
1. **Input:** Graph  $G = (V, E)$ . For each vertex  $v$ ,  $A(v)$  is list of its neighbors.
2.  $key(w)$  is the cost of the cheapest edge  $(v, w)$  with  $v \in T$ .  $blue(v, w)$  is the identity of this edge.
3. **Initialize:**
  - $key(v) = \infty$ , for all  $v \in V$ .
  - $H = makeHeap()$ .
  - $v = s$ .
4. **while**  $v \neq NIL$  **do**
  - **for**  $w \in A(v)$
  - **if**  $cost(v, w) < key(w)$
  - $key(w) = cost(v, w)$
  - $blue(w) = (v, w)$
  - **if**  $w \notin H$   $insert(w, H, key(w))$
  - **else**  $DecreaseKey(w, H, key(w))$
  - **end for**
  - $v = ExtractMin(H)$ .

# Illustration

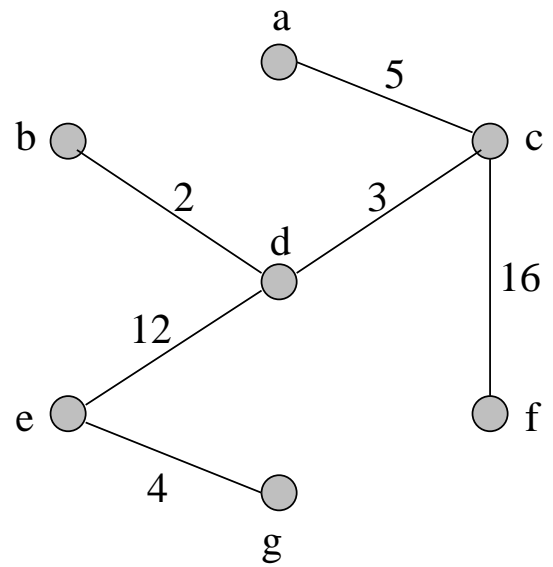
---



After 3 steps



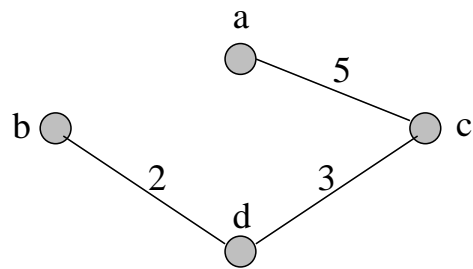
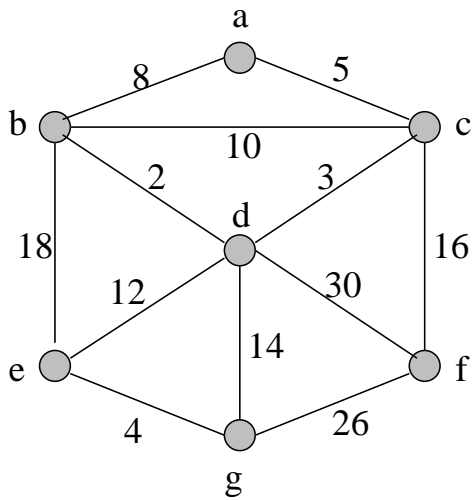
After 4 steps



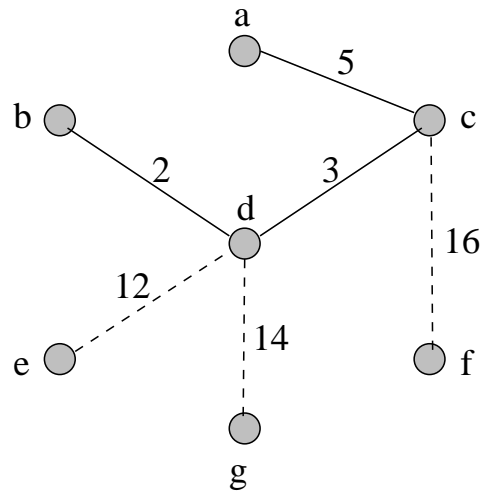
Final MST

# Illustration

---



After 3 steps



Blue edges and keys after 3 steps

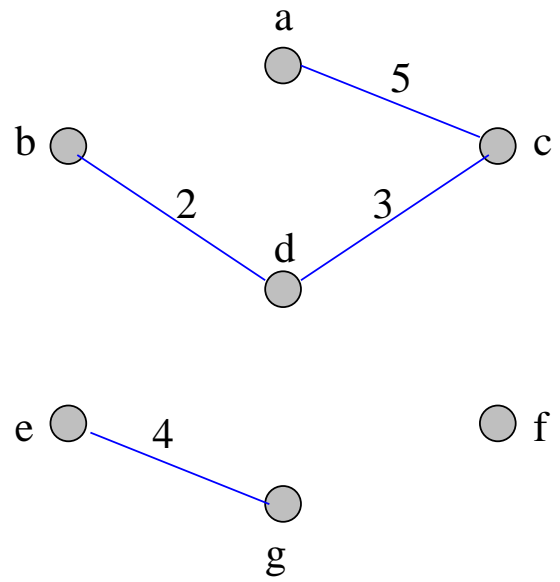
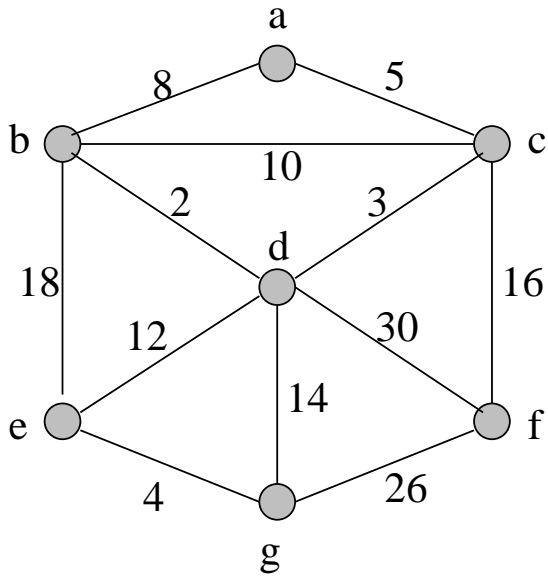
# Kruskal's Algorithm

---

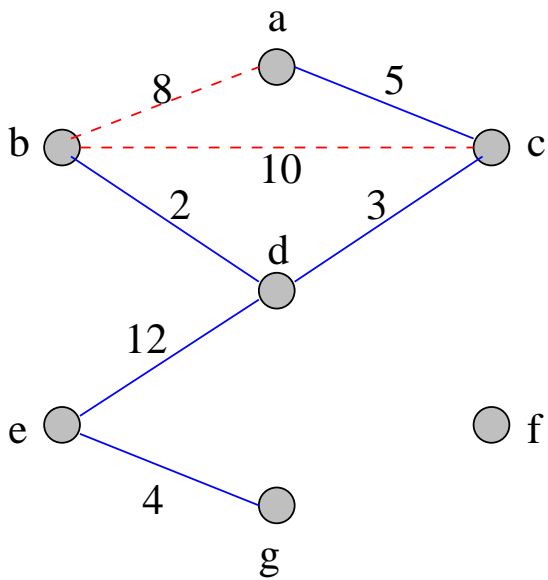
- Initially make each node of  $V$  a singleton tree.
- Scan edges of  $E$  in non-decreasing order of cost.
- scan edge  $e$ :  

If both endpoints of  $e$  in the same tree, color it **red**. Otherwise color  $e$  **blue**, and merge the two trees.

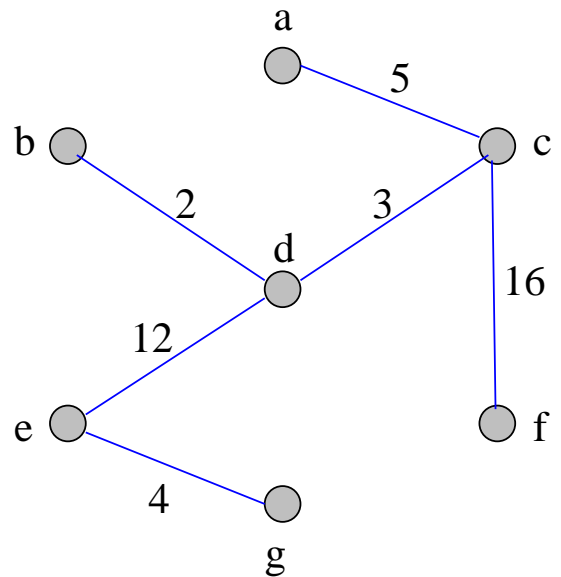
# Illustration



After 4 edges scanned



8, 10 colored red. 12 colored blue



Final MST

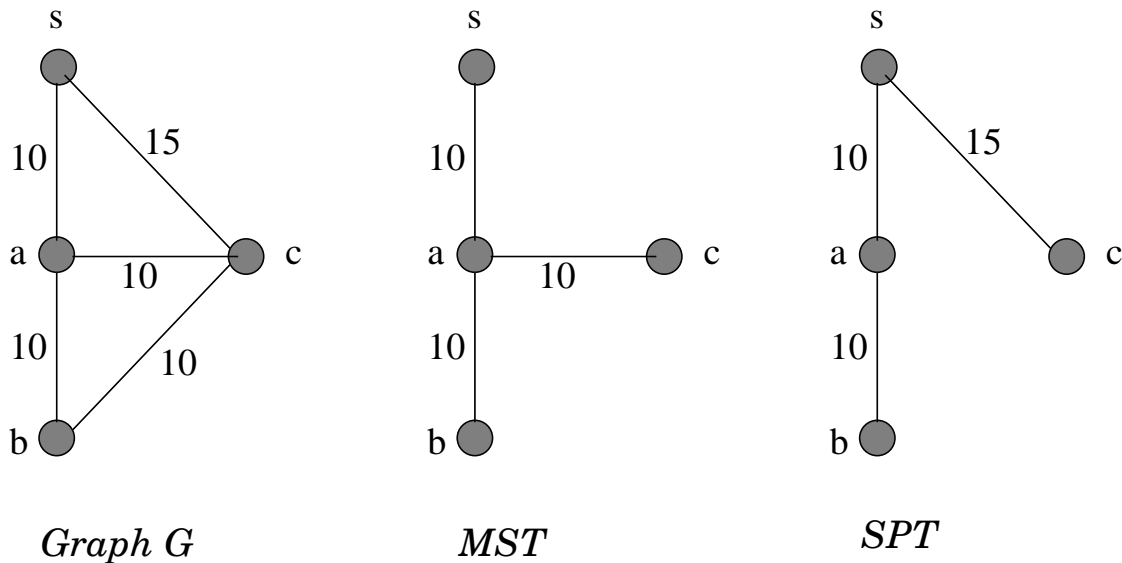
# Complexity Results

---

- **Heap implementation of Prim:**  $O(m \log_2 n)$ .
- **$d$ -Heap:**  $O(nd \log_d n + m \log_d n)$ .
- **Fibonacci Heap:**  $O(m + n \log n)$ .
- **Kruskal with Union-Find:**  
 $O(m \log n + m\alpha(n))$ .
- **Round-Robin:**  $O(m \log \log n)$ .
- **Latest theoretical bound:**  $O(m\alpha(n))$ .

# Balancing MST and SP

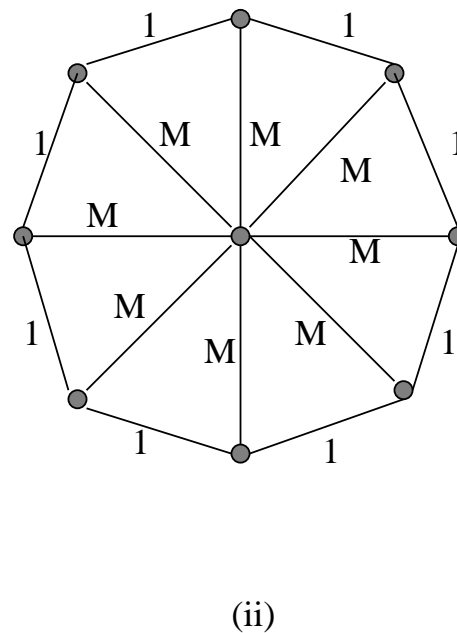
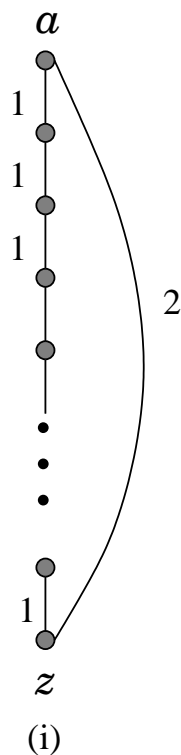
---



- MST minimizes *total* interconnection cost.
- SPT minimizes *individual* path lengths, from a root source.
- How does each do with the other cost metric?
- In Fig. 2,  $d_{mst}(a, c) = 20$ , but  $d_{spt}(a, c) = 15$ .
- In Fig. 3,  $cost(spt) = 35$ , but  $cost(mst) = 30$ .

# A Pathological Example

---



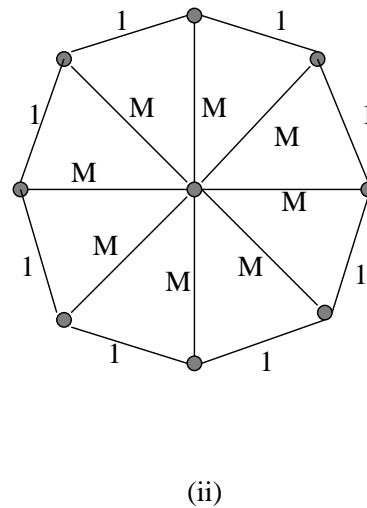
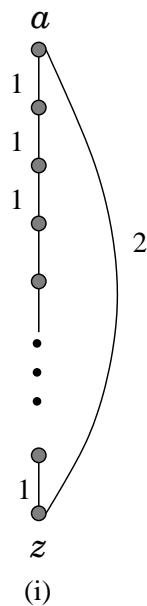
- **Fig. (i)** shows an example where  $d_{mst}(a, z) = n$ , while  $d_{spt} = 2$ .
- As  $n \rightarrow \infty$ , this ratio is unbounded.
- **Fig. (ii)** shows an example where  $cost(spt) = nM$ , while  $cost(mst) = n - 1 + M$ .
- As  $M \rightarrow \infty$ , this ratio is unbounded.

# Balanced Tree Theorem

---

**Theorem:** Pick any constant  $\alpha > 1$ , and let  $\beta = 1 + \frac{2}{\alpha-1}$ . Given  $G = (V, E)$  and a node  $s$ , there always exists a spanning tree  $T$  rooted at  $s$  such that

- $d_T(s, v) \leq \alpha \cdot d_{spt}(s, v)$ , for any  $v \in V$ .
- $cost(T) \leq \beta \cdot cost(MST)$ .



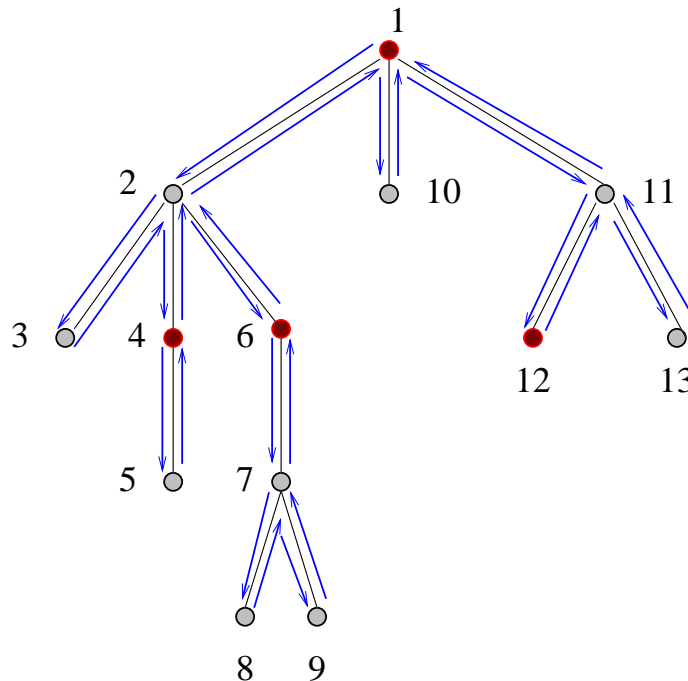
- Which  $T$  for these pathological cases?

# Pre-Order Numbering

---

**Pre-Order Lemma:** *Let  $T$  be a spanning tree with root  $s$ . Let  $z_0, z_1, \dots, z_k$  be any  $k + 1$  nodes listed in their pre-order sequence. Then,*

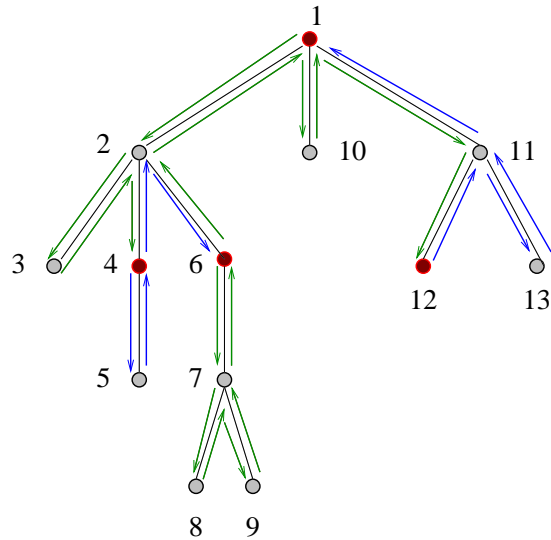
$$\sum_{i=1}^k d_T(z_{i-1}, z_i) \leq 2\text{cost}(T).$$



- Draw  $T$  in the plane. Let  $W$  be the doubling “walk” around  $T$  (pre-order).
- Each edge visited twice,  $\text{cost}(W) = 2\text{cost}(T)$ .

# Proof

---



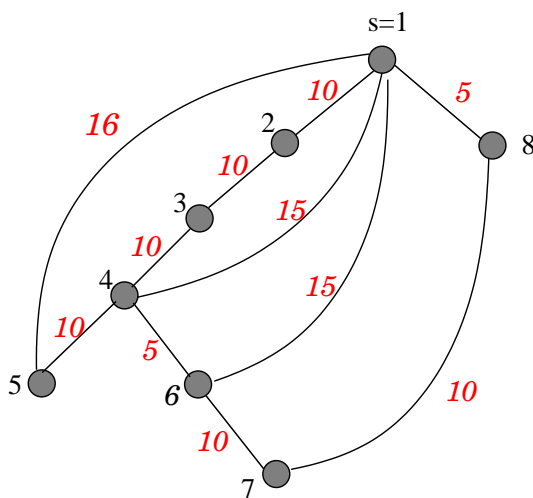
- $d_T(u, v)$  is tree path length.
  - Due to pre-ordering, **first** occurrence of  $z_{i-1}$  is **before** that of  $z_i$ .
  - Mark off portions of  $W$  that join first occurrence of  $z_{i-1}$  to that of  $z_i$ .
  - No edge of  $W$  marked more than once.
- Thus,

$$\sum_{i=1}^k d_T(z_{i-1}, z_i) \leq \text{cost}(W) \leq 2\text{cost}(T).$$

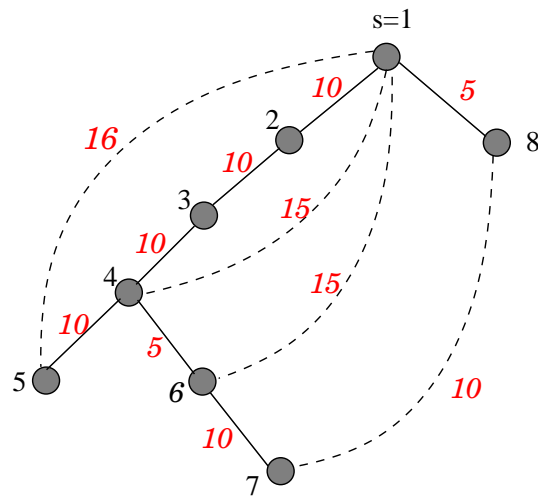
# Algorithm

---

- $\beta = 1 + \frac{2}{\alpha-1}$ . E.g.  $\alpha = 2, \beta = 3$ .
- Compute MST and its pre-order numbers, starting with  $s$ .
- Compute  $d_{spt}(s, v)$ , for all  $v$ .
- Initialize  $H = MST$ .
- for each node  $v$  in pre-order do  
    if  $d_H(s, v) > \alpha \cdot d_{spt}(s, v)$  then  
        add to  $H$  all edges of path  $P_{spt}(s, v)$ .
- Output SPT of  $H$ , rooted at  $s$ .

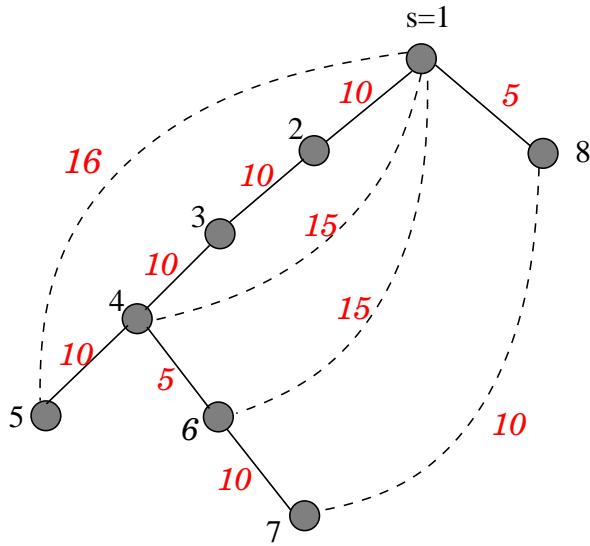


Graph G

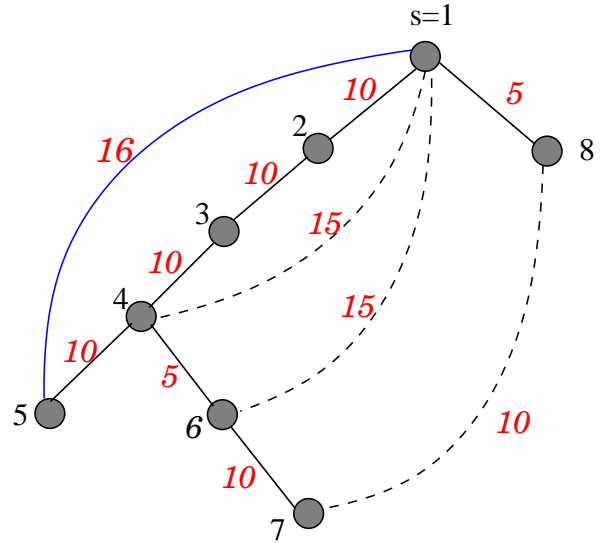


MST

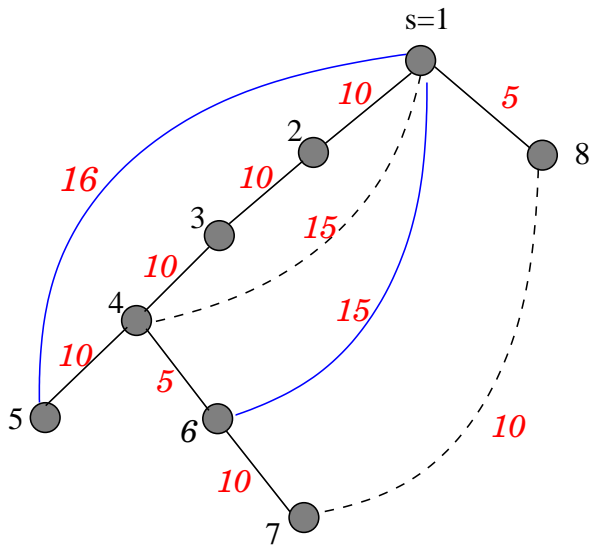
# Illustration



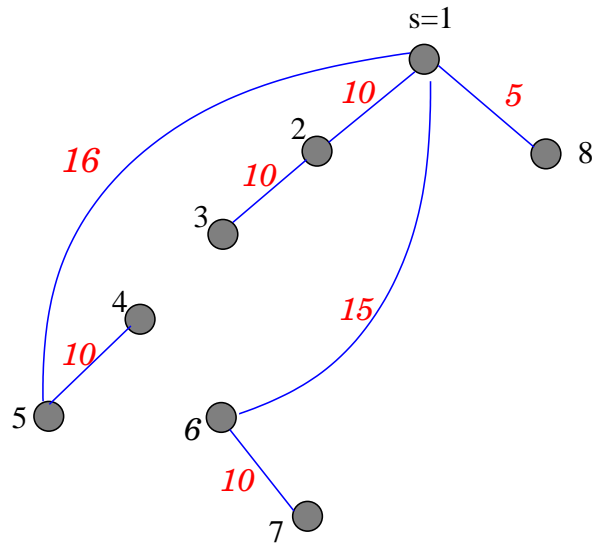
*MST*



*(1,5) added*



*(1,6) added*



*Final tree T output*

# Analysis

---

- $T$  satisfies  $d_T(s, v) \leq \alpha \cdot d_{spt}(s, v)$ ; algorithm adds SP whenever needed.
- Let  $z_0, z_1, \dots, z_k$  be the pre-order sequence of vertices that caused SP edges to be added to  $H$ .
- When  $z_i$  is examined,  $H$  contains the shortest path to  $z_{i-1}$ . So, we must have

$$d_{spt}(s, z_{i-1}) + d_{mst}(z_{i-1}, z_i) > \alpha \cdot d_{spt}(s, z_i).$$

- Sum these over all  $i$ :

$$\begin{aligned} \alpha d_{spt}(s, z_1) - d_{spt}(s, z_0) &< d_{mst}(z_0, z_1) \\ \alpha d_{spt}(s, z_2) - d_{spt}(s, z_1) &< d_{mst}(z_1, z_2) \\ &\vdots \\ \alpha d_{spt}(s, z_k) - d_{spt}(s, z_{k-1}) &< d_{mst}(z_{k-1}, z_k) \end{aligned}$$

- $\sum_{i=1}^k (\alpha - 1) d_{spt}(s, z_i) < \sum_{i=1}^k d_{mst}(z_{i-1}, z_i)$

# Analysis

---

- Thus, we have

$$\begin{aligned}\sum_{i=1}^k (\alpha - 1) d_{spt}(s, z_i) &< \sum_{i=1}^k d_{mst}(z_{i-1}, z_i) \\ &\leq 2 \text{cost}(MST)\end{aligned}$$

- Thus,

$$\sum_{i=1}^k d_{spt}(s, z_i) < \frac{2}{\alpha - 1} \text{cost}(MST)$$

- Since  $\text{cost}(H) = \text{cost}(MST) + \sum_{i=1}^k d_{spt}(s, z_i)$ , we get

$$\text{cost}(H) < \left(1 + \frac{2}{\alpha - 1}\right) \text{cost}(MST)$$

- Theorem proved.

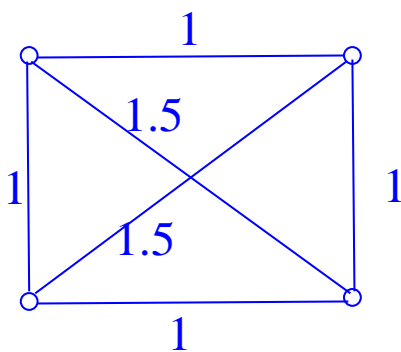
# Graph Spanners

---

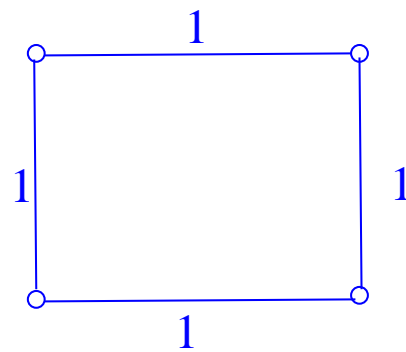
- A subgraph  $H = (V, E')$  is a  **$t$ -spanner** of graph  $G = (V, E)$ , where  $E' \subseteq E$ , if for all pairs  $u, v \in V$ ,

$$\text{dist}_H(u, v) \leq t \cdot \text{dist}_G(u, v)$$

- The parameter  $t$  is called the **stretch factor**.
- Total cost of  $H$  is the sum of costs of edges in  $H$ .
- Determine  $H$  with minimum total cost such that stretch factor is at most  $t$ .



Graph G

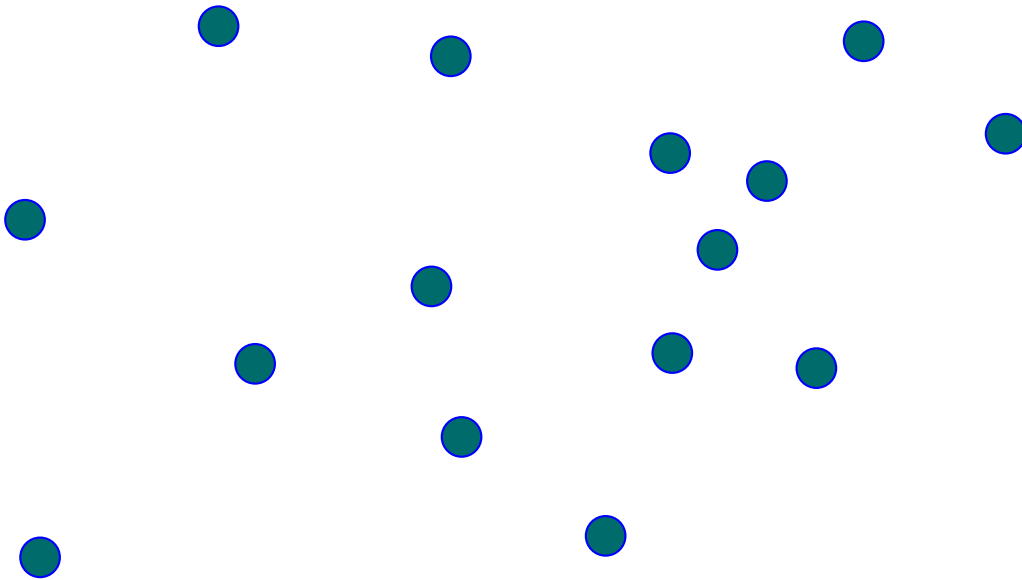


Spanner H

# Graph Spanners

---

- We want a small-size subgraph  $H$  that approximates well the shortest paths between all pairs of vertices in  $G$ .



- For instance, in building transportation networks, tradeoff between stretch factor and total network cost.
- What's the best  $H$  for this example, using Euclidean distances?

# Spanner Algorithm

---

- The following simple, elegant algorithm works.

**Spanner**  $(G, t)$

$H \leftarrow (V, \emptyset);$

Sort edges of  $G$  in increasing cost order;

Let  $e_1, e_2, \dots, e_m$  be this order;

**for**  $i = 1$  **to**  $m$  **do**

    Let  $e_i = (u, v);$

**if**  $\text{dist}_H(u, v) > t \cdot w(e_i)$

$H \leftarrow H \cup \{e_i\};$

**end**

**end**

**Output**  $H;$

# Correctness

---

**Lemma:** The subgraph  $H$  output by Spanner is a  $t$ -spanner of  $G$ .

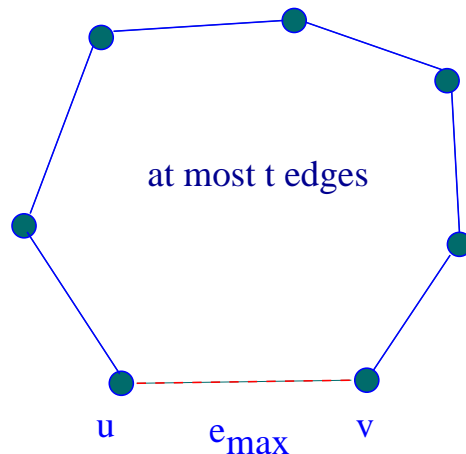
- Let  $P(u, v)$  be the shortest path in  $G$ .
- Consider an edge  $e$  on this path.
- $e$  was either added to  $H$  by spanner algorithm, or there is a path of length  $t \cdot w(e)$  in  $H$ .
- Concatenating these approximate paths in  $H$ , we get a path in  $H$  whose length is at most  $t \cdot P(u, v)$ .

# Correctness

---

**Lemma:** If  $C$  is a cycle in  $H$ ,  $size(C) > t + 1$ .

- Suppose  $\exists$  a cycle  $C$  with  $\leq t + 1$  edges.
- Let  $e_{\max} = (u, v)$  be its most costly edge.
- When algorithm scans  $e_{\max}$ , all other edges of  $C$  have been added.



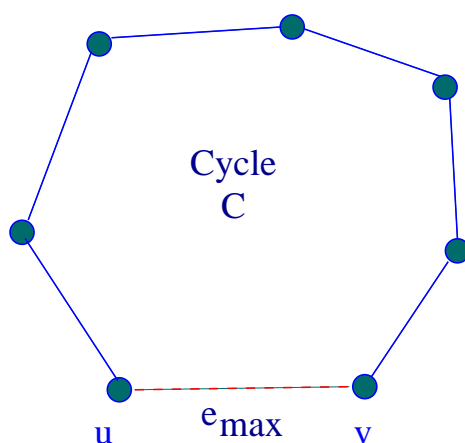
- Thus, there is already a path from  $u$  to  $v$  of cost  $\leq t \cdot e_{\max}$  (each of the  $t$  edges is  $\leq w(e_{\max})$ ).
- So, edge  $e_{\max}$  will not be added.

# Correctness

---

**Lemma:**  $w(C - e) > t \cdot w(e)$ , for any  $e \in C$ .

- Suppose  $\exists$  a cycle  $C$  and edge  $e$  with  $w(C - e) \leq t \cdot w(e)$ .



- So,  $w(C) \leq (t + 1)w(e) \leq (t + 1)w(e_{\max})$ .
- When considering  $e_{\max}$ , it won't be added!

# Correctness

---

**Lemma:** *MST* contained in *H*.

- Straightforward. Prove it yourself.
- The key is to show

$$w(E_v) \leq \frac{2w(MST)}{t-1}$$

where  $E_v$  is edges of  $H - MST$  incident to vertex  $v$ .

- This is sufficient to prove

$$\begin{aligned} w(H) &\leq w(MST) + \frac{1}{2} \sum_v w(E_v) \\ &\leq w(MST) \left( 1 + \frac{n}{t-1} \right) \end{aligned}$$

# Proof

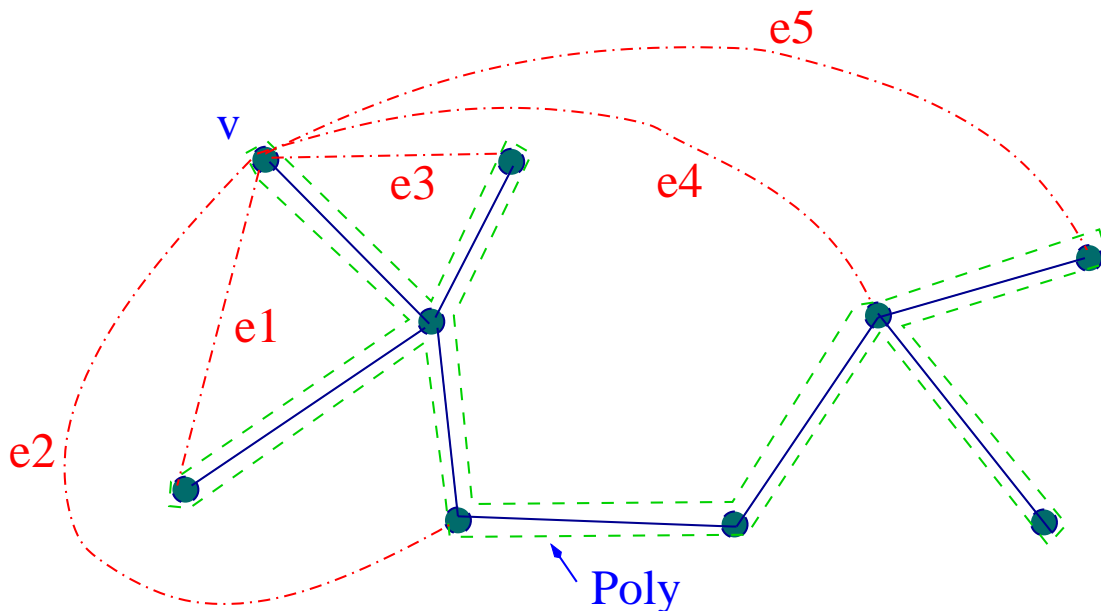
---

**Theorem:**  $w(E_v) \leq \frac{2w(MST)}{t-1}$ .

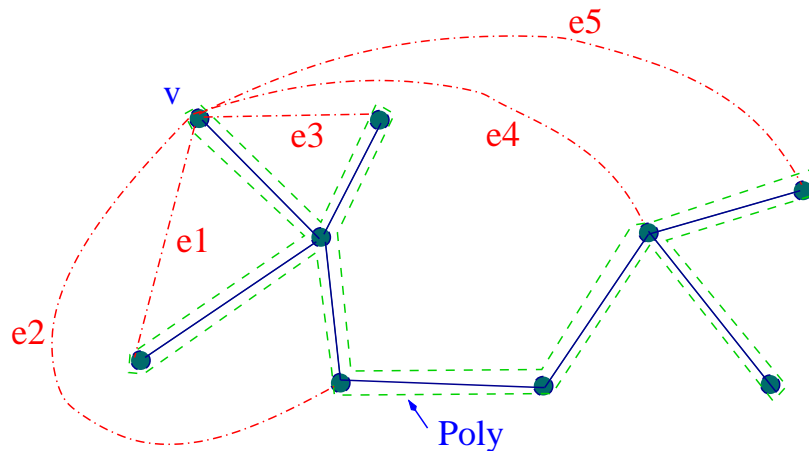
- Let  $e_1, e_2, \dots, e_k$  be edges in  $E_v$ .
- For any edge  $e \in H$ , and **any** path  $P$  in  $H$  joining the endpoints of  $e$ , we have

$$t \cdot w(e) < w(P)$$

- Let **Poly** be the polygon defined by doubling MST.



# Proof



- $W_i$  is perimeter of Poly after  $e_1, \dots, e_i$  added. Initially,  $W_0 = MST$ .
- What happens when  $e_i = (u, v)$  is added?

$$\begin{aligned} W_i &= W_{i-1} + w(e_i) - w(P(u, v)) \\ &< W_{i-1} - w(e_i) \cdot (t - 1) \end{aligned}$$

$$(t - 1)w(e_i) < W_{i-1} - W_i$$

- Thus,  $(t - 1) \sum_{i=1}^k w(e_i) \leq W_0 - W_k$ .
- Since  $W_0 = 2w(MST)$  and  $W_k > 0$ ,

$$E_v = \sum_{i=1}^k w(e_i) < \frac{2w(MST)}{t - 1}.$$

# Spanner Results

---

- Althofer et al. result gives spanners with  $O(n^{1+\frac{2}{t-1}})$  edges.
- Peleg-Schaffer shows that there are  $n$ -vertex graphs, whose all  $t$ -spanners contain  $\Omega(n^{1+\frac{1}{t}})$  edges.
- Much better results know if distances are Euclidean.
- In Euclidean  $k$ -space (constant  $k$ ) spanners with  $O(n)$  edges exist for any constant stretch factors  $t$ .
- Such spanners possible with  $O(MST)$  weight,  $O(1)$  degree,  $O(\log n)$  diameter.