Communications Network Design

lecture 16

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

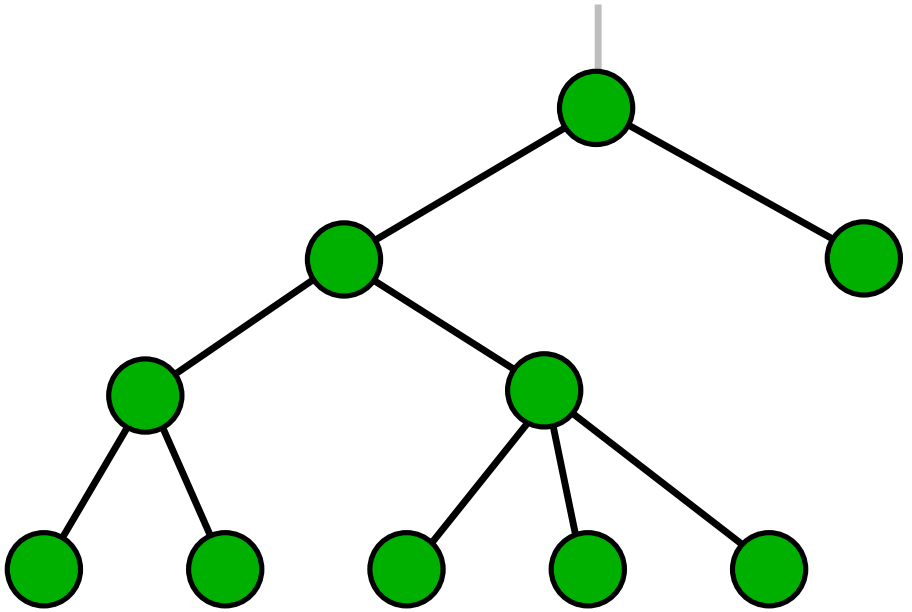
Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

May 20, 2009

Tree-like networks

Tree-like networks, and algorithms for their design: minimum spanning tree problem, spanning trees and spanning tree protocol, greedy methods (Kruskal's and Prim's methods).

Tree-like networks



A Tree

- connected
- does not contain any cycles (loops)
  - A graph contains no cycles if there is no path of non-zero length $\{v_i\}_{i=1}^{k}$ through the graph such that $v_0 = v_k$

Spanning Tree

Spanning tree, alternative definitions

- a tree that connects all nodes in the graph
- connected graph where the number of links in $G(N,T)$ is $|T| = |N-1|$
- the graph is connected, but if we ommit a single link, it becomes disconnected
- every pair of vertices is connected along one and only one path

Given $|N|$ nodes,

- there are as many as $|N|^{|N|-2}$ such trees
- more even than the number of paths

Spanning Tree

- example application: cable TV network
  - need to get a broadcast TV signal from one (root) to many (leaves)
  - maybe we want to do this as cheaply as possible?
- example application: Ethernet
  - Spanning Tree Protocol
- example application: Fibre-To-The-Node (FTTN)
  - proposed design for Australian broadband
  - hybrid fibre/copper network
    - use copper telephone lines from home
    - run fibre out to "nodes"
    - result is a tree-like network
  - where should nodes be?

Spanning Tree Protocol

- an Ethernet can have multiple possible switching paths
  - for reliability
- but Ethernet effectively broadcasts some messages
  - e.g. ARP
  - route loops could be REALLY bad
  - looping broadcasts would take up all available bandwidth
  - no mechanism at the IP layer can stop this, as it is happening at layer 2
- Spanning Tree Protocol (STP) intended to create a tree
  - hence avoid loops
- Is the STP optimizing anything we care about?

Spanning Tree Protocol

- Two versions of STP
  - DEC and IEEE (not compatible) we will look at IEEE
- switches are assigned numerical **priority**
- Ethernet switch with lowest priority is **root**
  - tie break is lowest MAC address
    - MAC addresses are unique
    - combination of priority and MAC is called **node ID**
- each Ethernet switch port is given a **cost**
  - based on bandwidth of the link
    - see next slide
  - like a link weight (in Dijkstra)

Port costs

- default costs based on bandwith
  - old version based on 1 Gbps/(link bandwidth)
  - new version arbitrary table
    - port cost has to be integer > 0
    - new costs account for link speeds > 1 Gbps

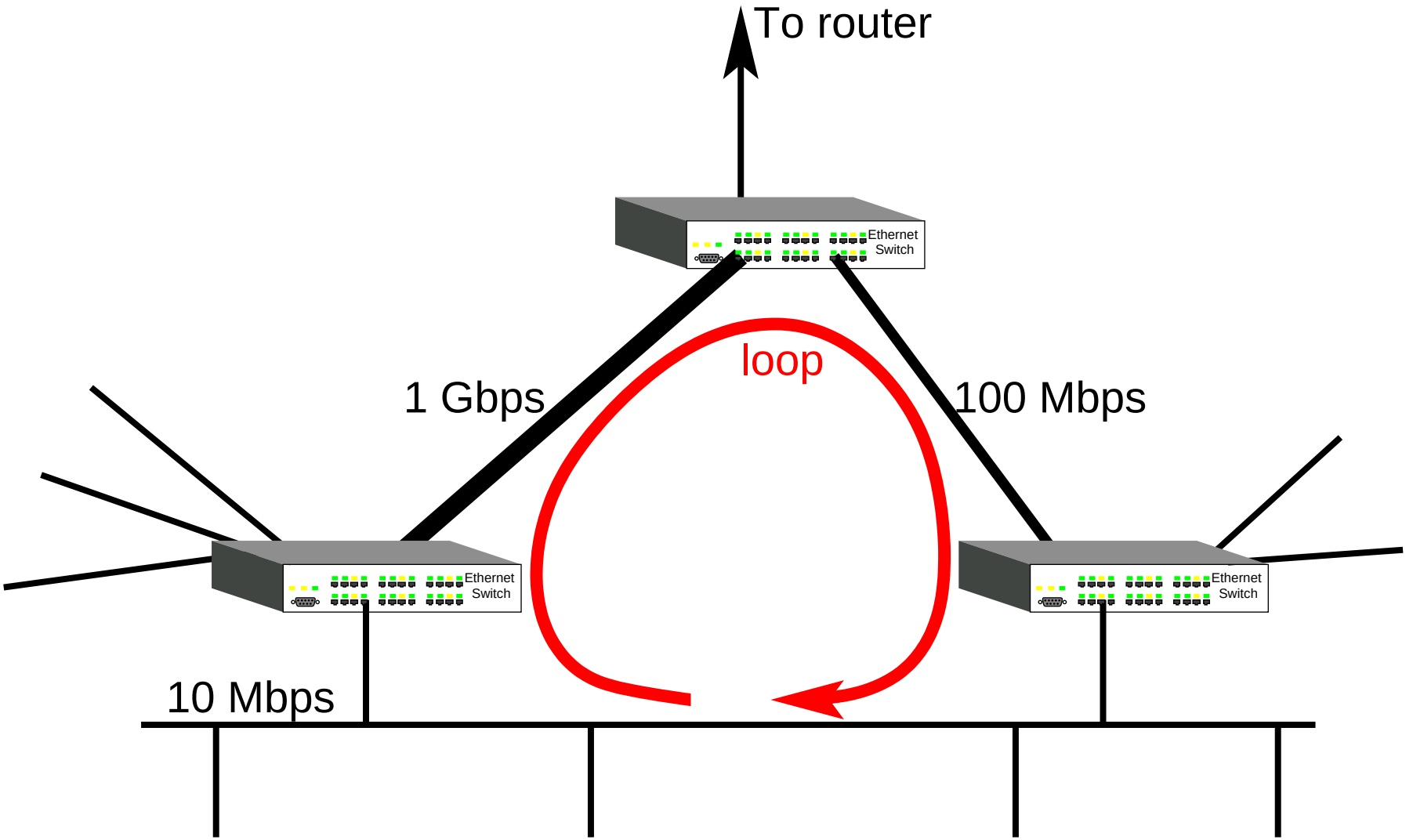
Media	new cost	old cost
10 Mbps	100	100
100 Mbps	19	10
1 Gbps	4	1
10 Gbps	2	1

- gives port ID
  - MAC address is used as a tie break again

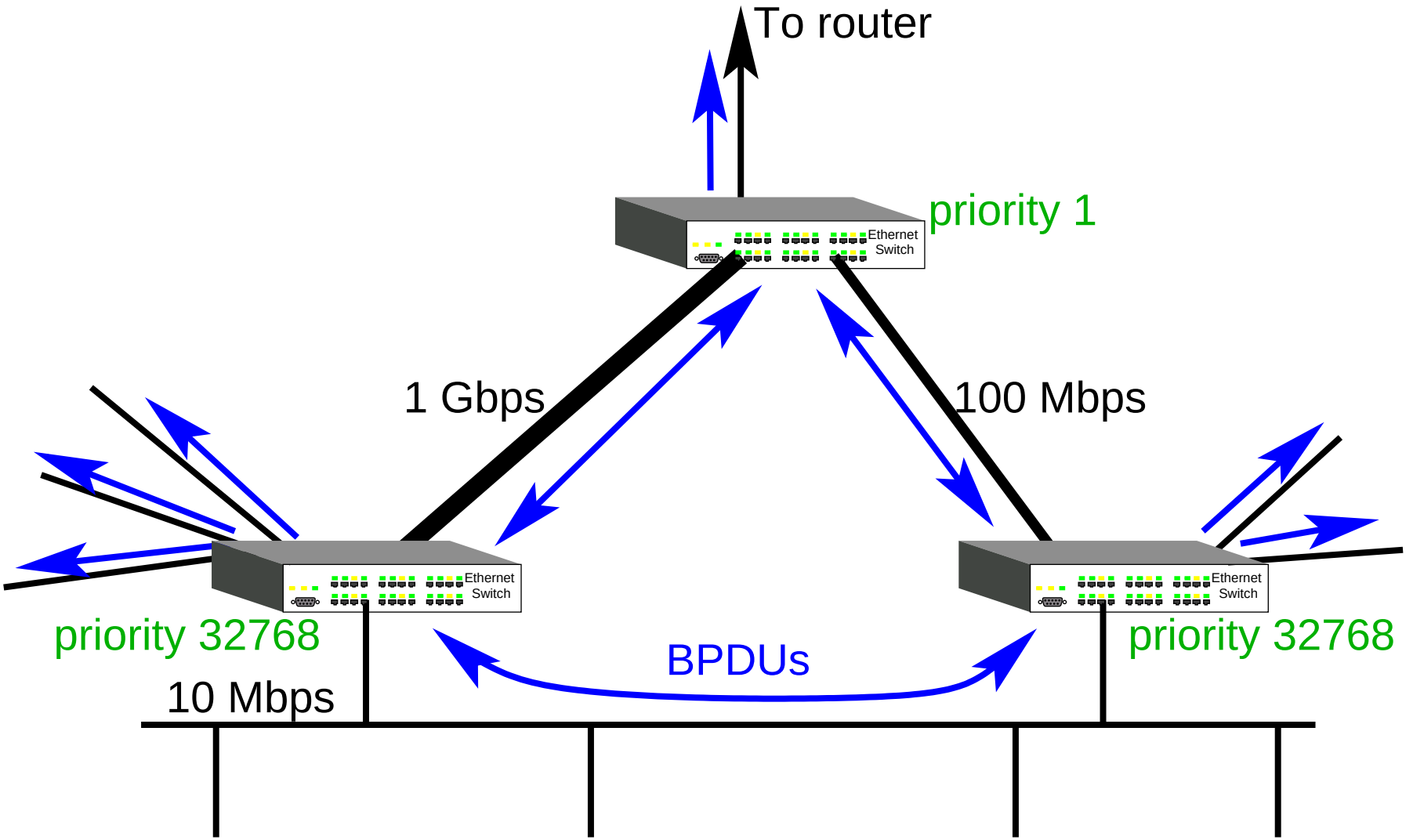
Spanning Tree Protocol

- Ethernet switches send Bridge Protocol Data Unit (BPDU)
  - e.g. Cisco default is every 2 seconds
  - contains information above, and state
  - flooded through network
- switches try to find cheapest path to the root switch
- closest port to the root on a switch is called the **root port**
- each Ethernet segment chooses the port advertizing the shortest path to the root
  - label the switch on this path the **designated switch**
- ports not on the designated switch are **blocked**
  - put into backup mode
  - they still listen to BPDUs, but don't forward packets

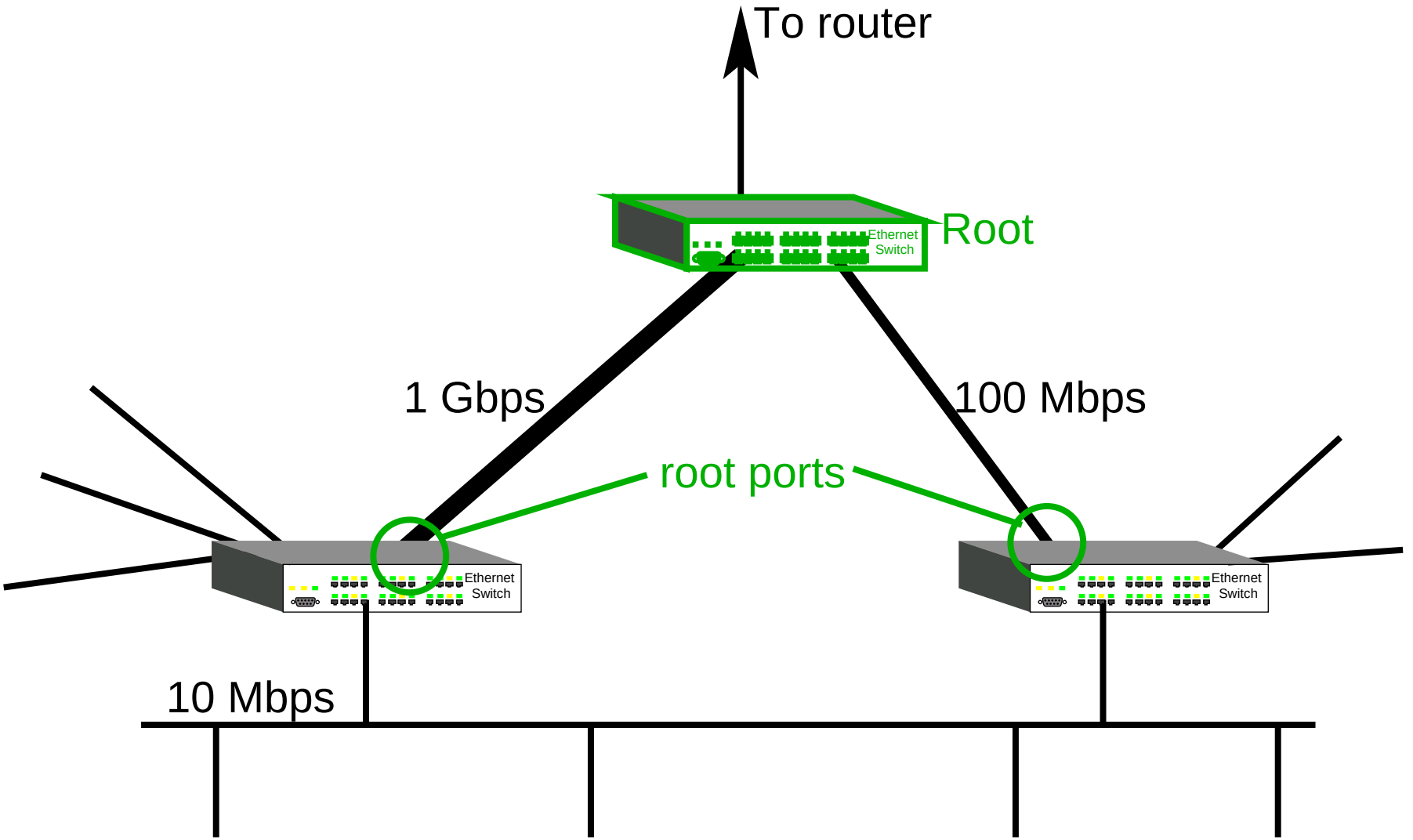
Spanning Tree Protocol Example



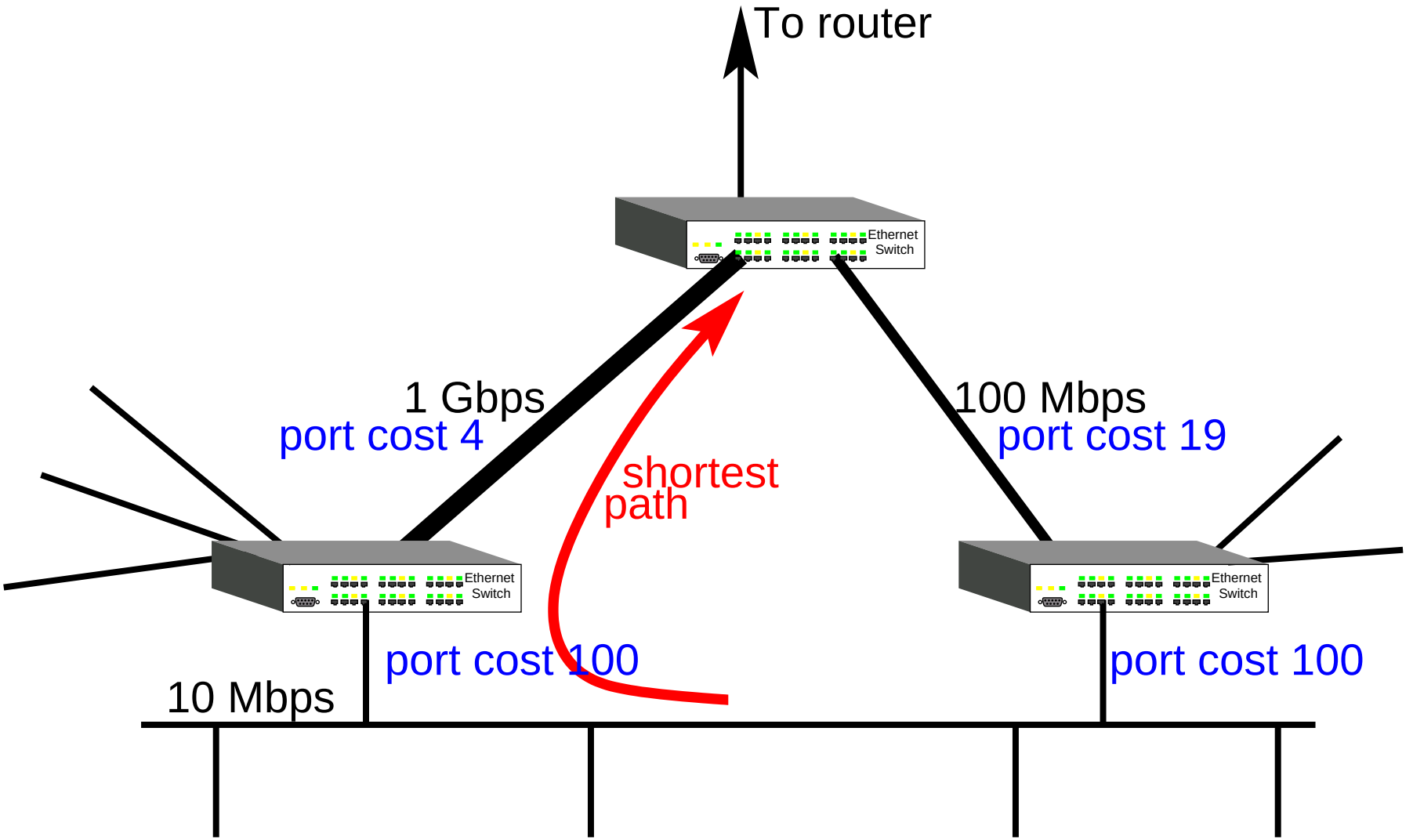
Spanning Tree Protocol Example



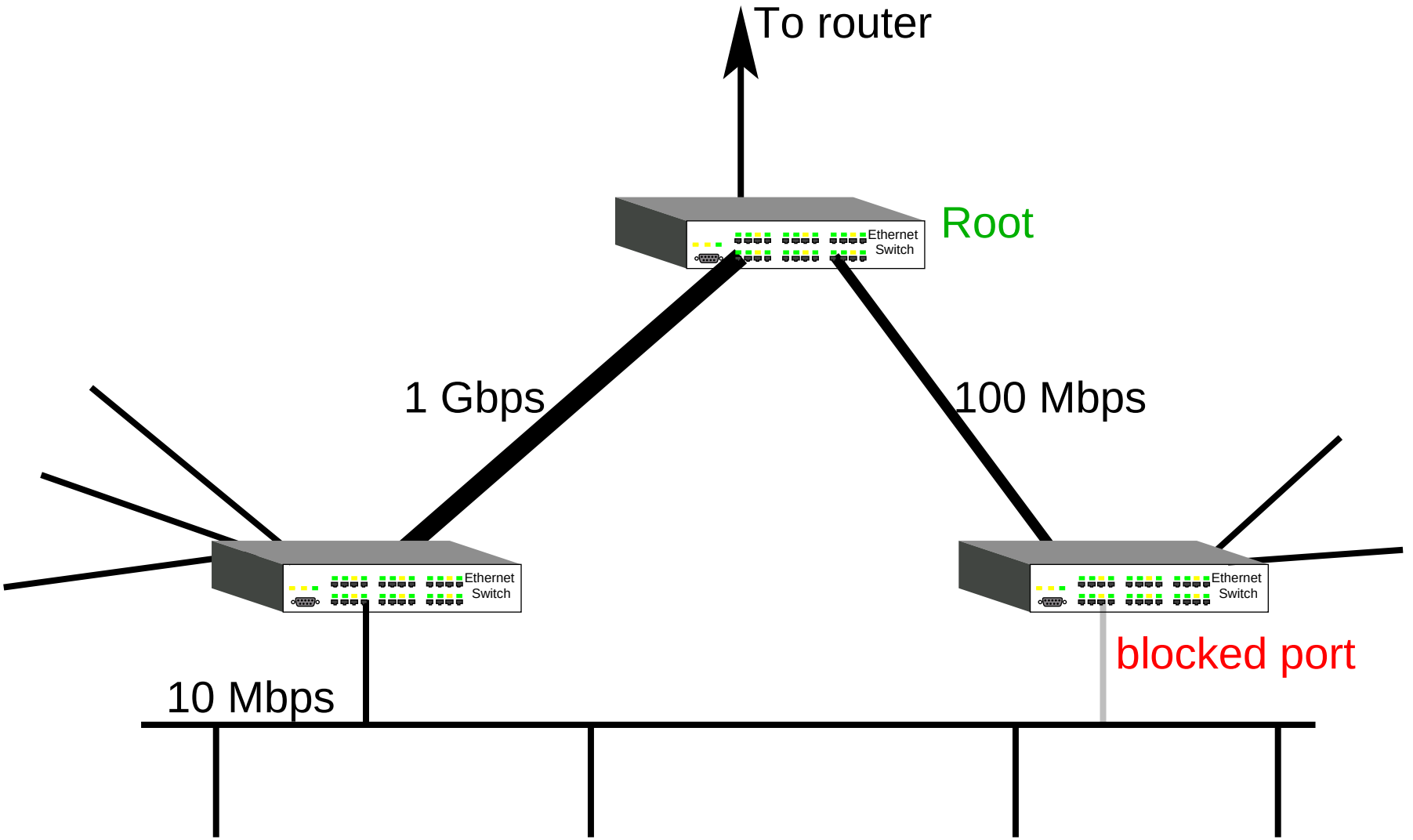
Spanning Tree Protocol Example



Spanning Tree Protocol Example



Spanning Tree Protocol Example



Algorhyme

Algorhyme

I think that I shall never see
A graph as lovely as a tree.
A tree which must be sure to span.
So packets can reach every LAN.
First the root must be selected.
By ID, it is elected.
Least cost paths from Root are traced.
In the tree these paths are placed.
A mesh is made by folks like me.
Then bridges find a spanning tree.

STP is attributed to Radia Perlman as is the poem above.

Minimum Spanning Trees

STP is doing an optimization:

- it is minimizing the path length for each leaf to get to the root
- using somewhat arbitrary link weights!
- effectively it is trying to minimize congestion by switching traffic on higher bandwidth paths

Another standard optimization is to minimize the total cost of the tree

- the exact problem depends on what we mean by "costs"
- often referred to as Minimum Spanning Tree (MSP)

Minimum Spanning Trees

Take a general linear cost model

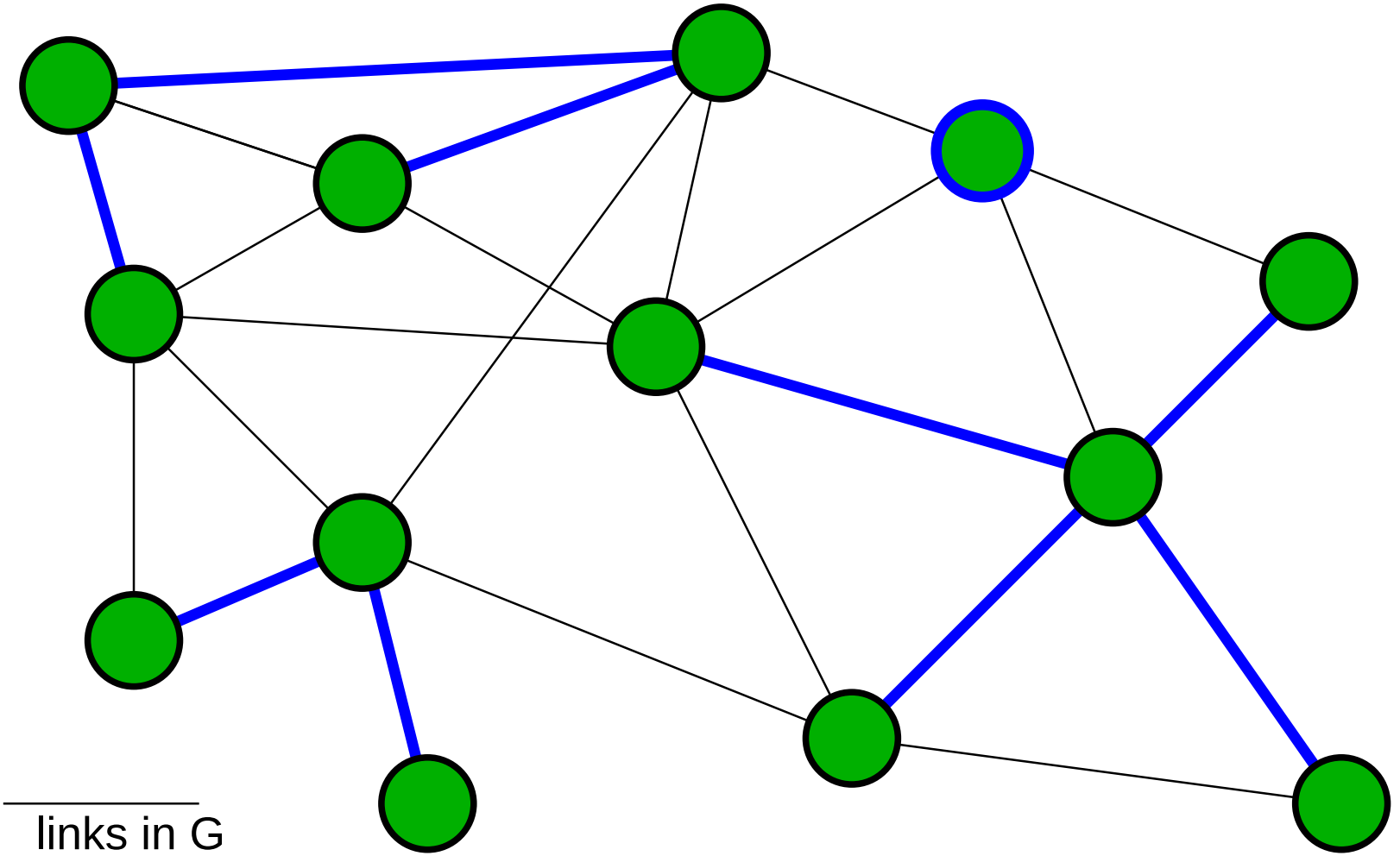
$$C(\mathbf{f}) = \sum_{e \in L} (\alpha_e f_e + \beta_e)$$

- various subcases exist
- **minimum weight spanning tree** (MWST)
  $\alpha_e = 0$

$$C(\mathbf{f}) = \sum_{e \in L} \beta_e$$

Forests

A collection of sub-trees is called a **forest**



links in G

links in the forest

Properties of importance

Proposition 1: If $\mathcal{F}, \mathcal{G}$ are two forests for a network and $|\mathcal{F}| > |\mathcal{G}|$, then $\exists$ link $e \in \mathcal{F}$ such that $\mathcal{G} \cup \{e\}$ is a forest.

Proof: Suppose the sub-trees in the forest $\mathcal{G}$ are denoted $G_1, G_2, \ldots G_k$ (so each is a connected component of $\mathcal{G}$ and each has no cycles). Let F_i be the set of links in $\mathcal{F}$ which have **both** endpoints in component G_i of $\mathcal{G}$. Now since $\mathcal{F}$ is a forest, and $F_i \subseteq \mathcal{F}$, each F_i is a forest, so

$$\begin{aligned} \text{number of links in } F_i &\leq (\#\text{ of nodes in } F_i) - 1 \\ &\leq (\#\text{ of nodes in } G_i) - 1 \\ &= \#\text{ links in } G_i \\ &\quad (\text{because } G_i \text{ is a tree}) \end{aligned}$$

Properties of importance

Proof: (of prop. 1 continued)

$$\therefore |F_i| \leq |G_i|$$
$$\therefore \sum_{i=1}^{k} |F_i| \leq \sum_{i=1}^{k} |G_i| = |\mathcal{G}| < |\mathcal{F}|$$

Therefore there is a link in $\mathcal{F}$ which does not have both endpoints in the same component (G_i) of $\mathcal{G}$. Hence $\mathcal{G} \cup \{e\}$ will not contain a cycle. Therefore $\mathcal{G} \cup \{e\}$ is a forest.□

Properties of importance

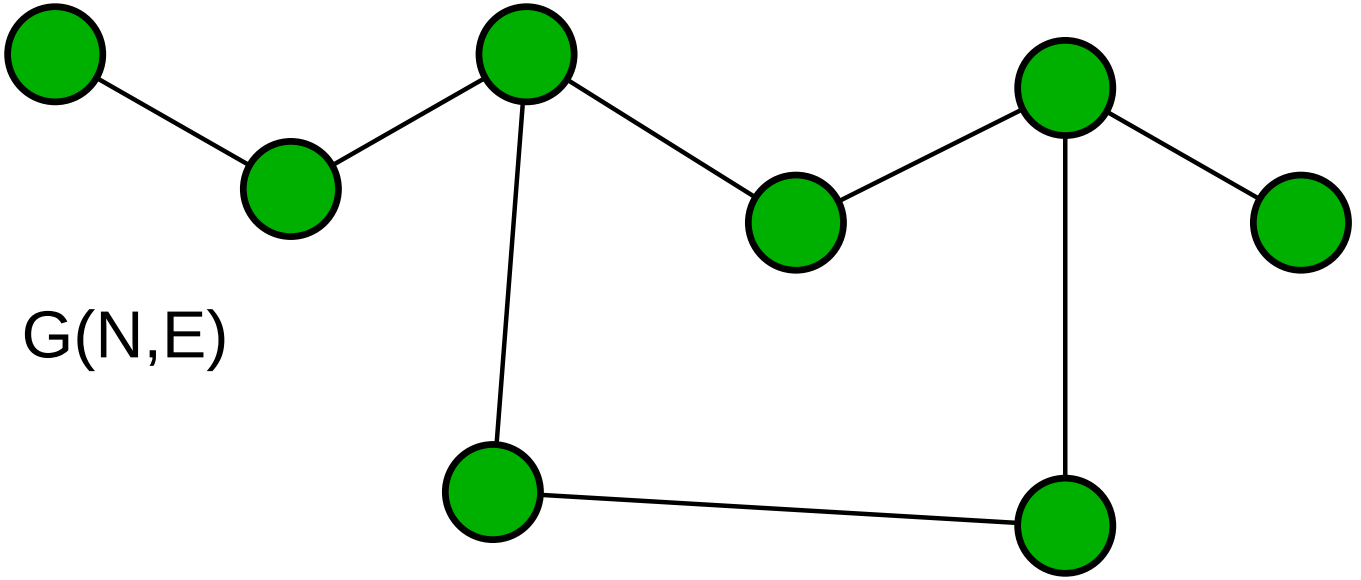
Proposition 2: Given a fragment F of an MWST, let e be a link of minimum weight with one node in F and the other not in F. i.e. e is such that $\beta_e = \min\{\beta_{ij} \mid i \in F, j \notin F\}$ Then $F \cup \{e\}$ is a fragment of an MWST

Proof: If $\nexists e$, then all nodes of G are in F, and therefore F is an MWST.

Suppose $\exists e = (i, j)$ such that $\beta_e = \min\{\beta_{ij} \mid i \in F, j \notin F\}$. Denote by T the minimum weight spanning tree of which F is a fragment. If $e \in T$, we are done, so assume $e \notin T$. Then $T \cup \{e\}$ has a cycle. Since node $j \notin F$, there is a link $e' \neq e$ which is <u>on</u> the cycle and on T, and has <u>one</u> node on F (see diagram).

Properties of importance

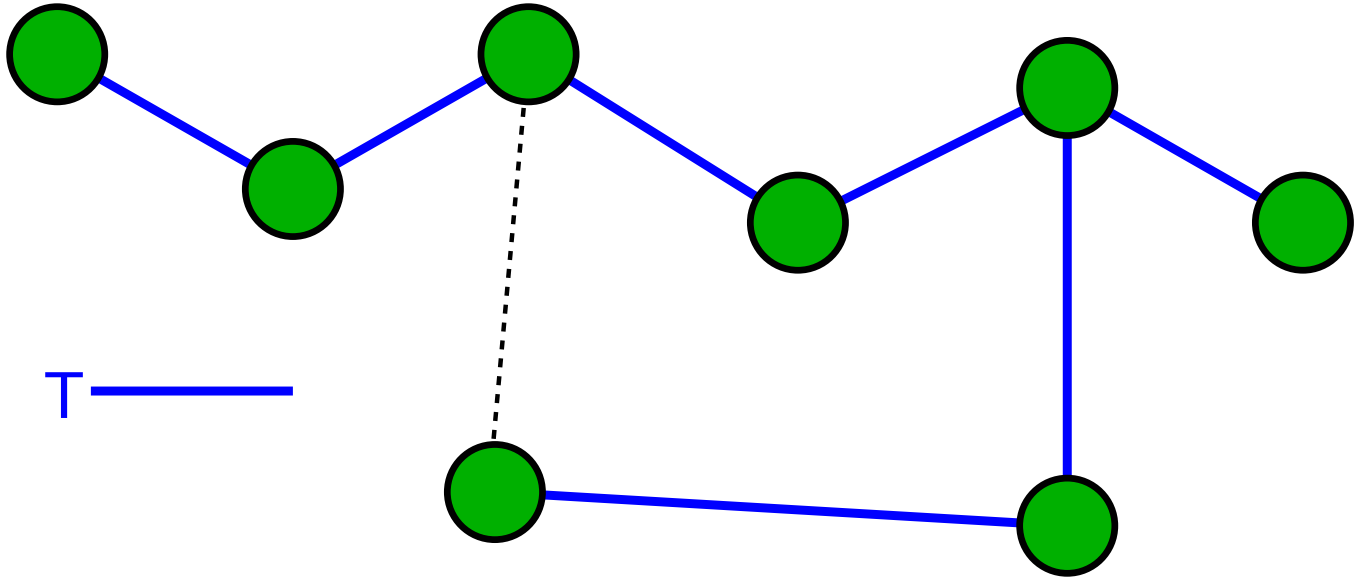
Proof: (of prop. 2 continued)



Delete e' from T and add e to T. We still have $|N|$ nodes and $|N|-1$ links. $\therefore$ we still have a spanning tree, T^* say. But $\beta_e \leq \beta_{e'}$ by definition of e. So T^* has a cost $\leq$ that of T. Therefore T^* is an MWST, and therefore $F \cup \{e\}$ is a fragment of an MWST.□

Properties of importance

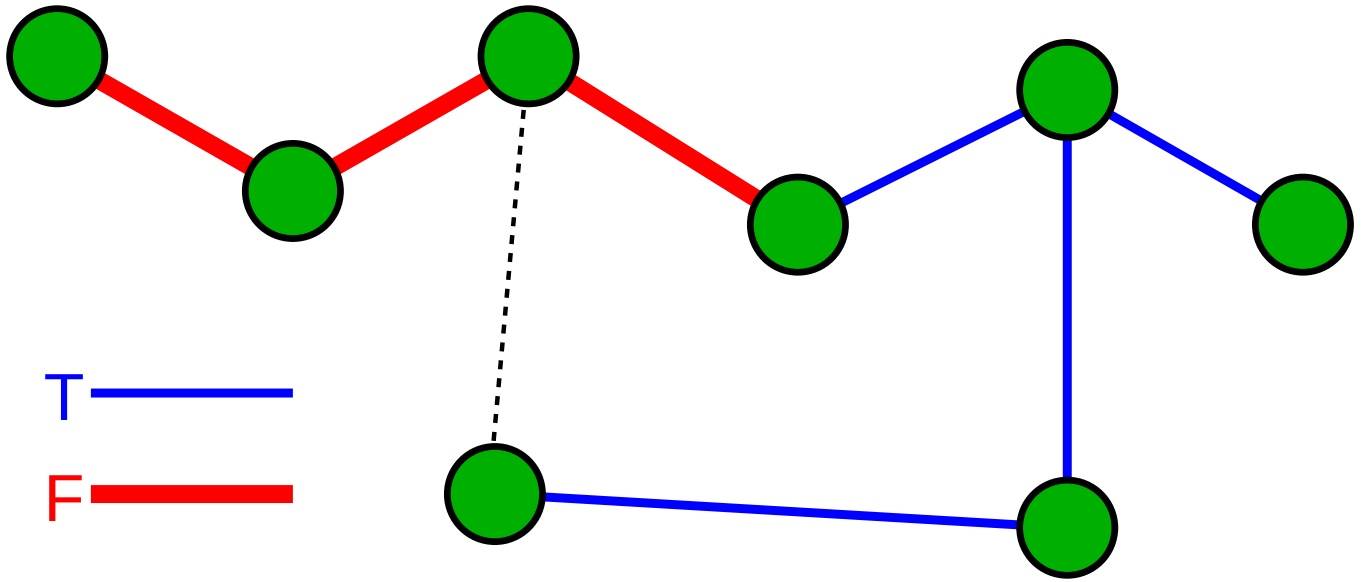
Proof: (of prop. 2 continued)



Delete e' from T and add e to T. We still have $|N|$ nodes and $|N|-1$ links. $\therefore$ we still have a spanning tree, T^* say. But $\beta_e \leq \beta_{e'}$ by definition of e. So T^* has a cost $\leq$ that of T. Therefore T^* is an MWST, and therefore $F \cup \{e\}$ is a fragment of an MWST.□

Properties of importance

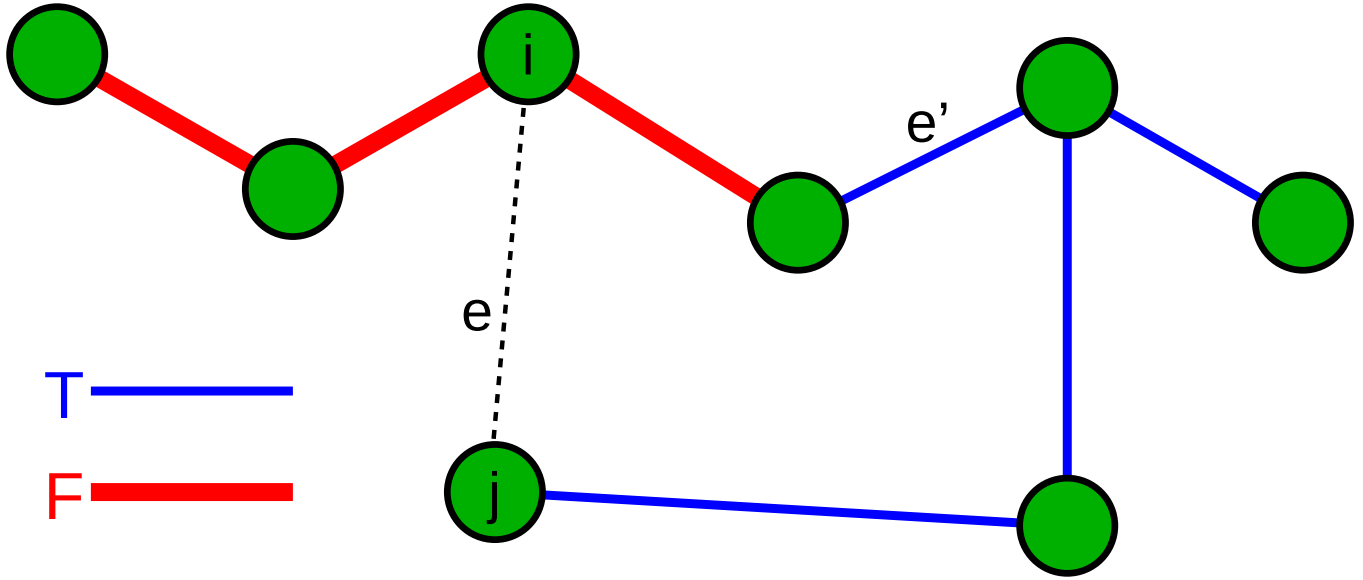
Proof: (of prop. 2 continued)



Delete e' from T and add e to T. We still have $|N|$ nodes and $|N|-1$ links. $\therefore$ we still have a spanning tree, T^* say. But $\beta_e \leq \beta_{e'}$ by definition of e. So T^* has a cost $\leq$ that of T. Therefore T^* is an MWST, and therefore $F \cup \{e\}$ is a fragment of an MWST.□

Properties of importance

Proof: (of prop. 2 continued)



Delete e' from T and add e to T. We still have $|N|$ nodes and $|N|-1$ links. $\therefore$ we still have a spanning tree, T^* say. But $\beta_e \leq \beta_{e'}$ by definition of e. So T^* has a cost $\leq$ that of T. Therefore T^* is an MWST, and therefore $F \cup \{e\}$ is a fragment of an MWST.□

Greedy methods

- at each step we make the best choice
  - don't ever go back
- e.g. Dijkstra, Minoux's greedy method
- advantage
  - generally pretty simple
- disadvantage
  - doesn't reach true optimum in many cases
    - results are still sometimes quite good
  - Dijkstra does find an optimum
- two new examples today
  - Kruskal and Prim's methods
  - both are optimal

Kruskal's method [1]

Input a connected network $G(N,E)$ with link weights $\beta_e \geq 0, \forall e \in E$

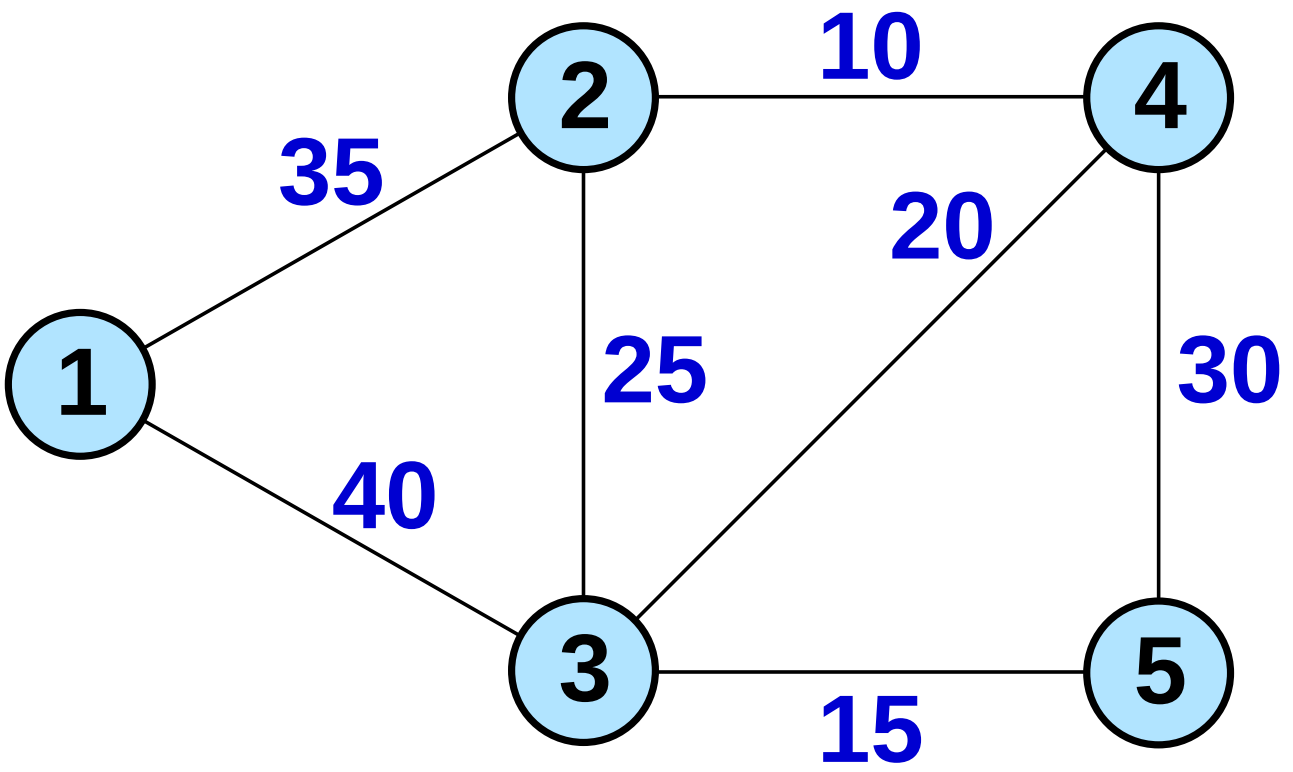
1. **Initialize:** list the links in increasing order of β_e, e.g.
$$\beta_{e_1} \leq \beta_{e_2} \leq \cdots \leq \beta_{e_{|E|}}$$
2. **While** not a spanning tree
   - choose the next link in the list
     - if it doesn't form a cycle, add it to the tree
     - if it does, then discard the link

Alternative step 2: for $i = 1, 2, \ldots, |E|$

- if adding link e_i would not create a cycle, add it to the tree, otherwise discard it

Kruskal's method example

Input graph with link weights β_e

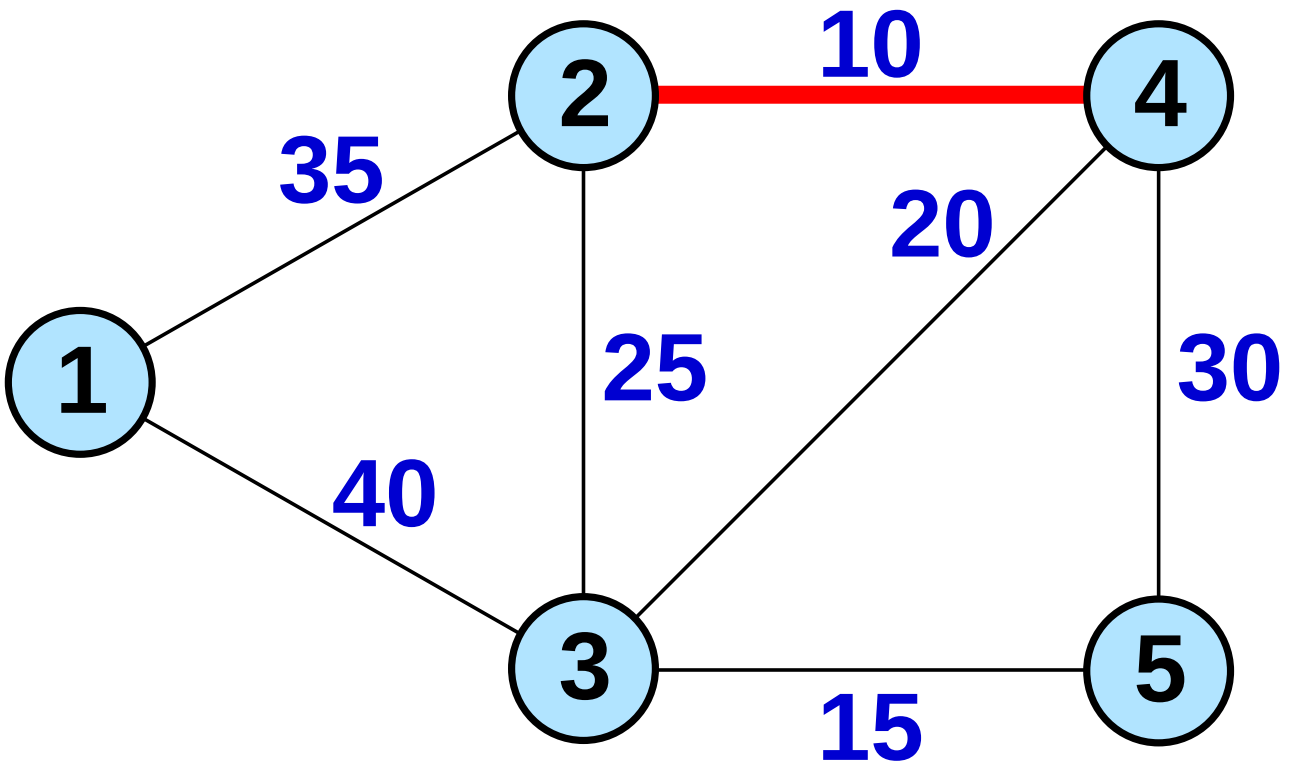


Obviously, the links in order of increasing β_e are:

link	$(2,4)$	$(3,5)$	$(3,4)$	$(2,3)$	$(4,5)$	$(1,2)$	$(1,3)$
$\beta_e =$	10	15	20	25	30	35	40

Kruskal's method example

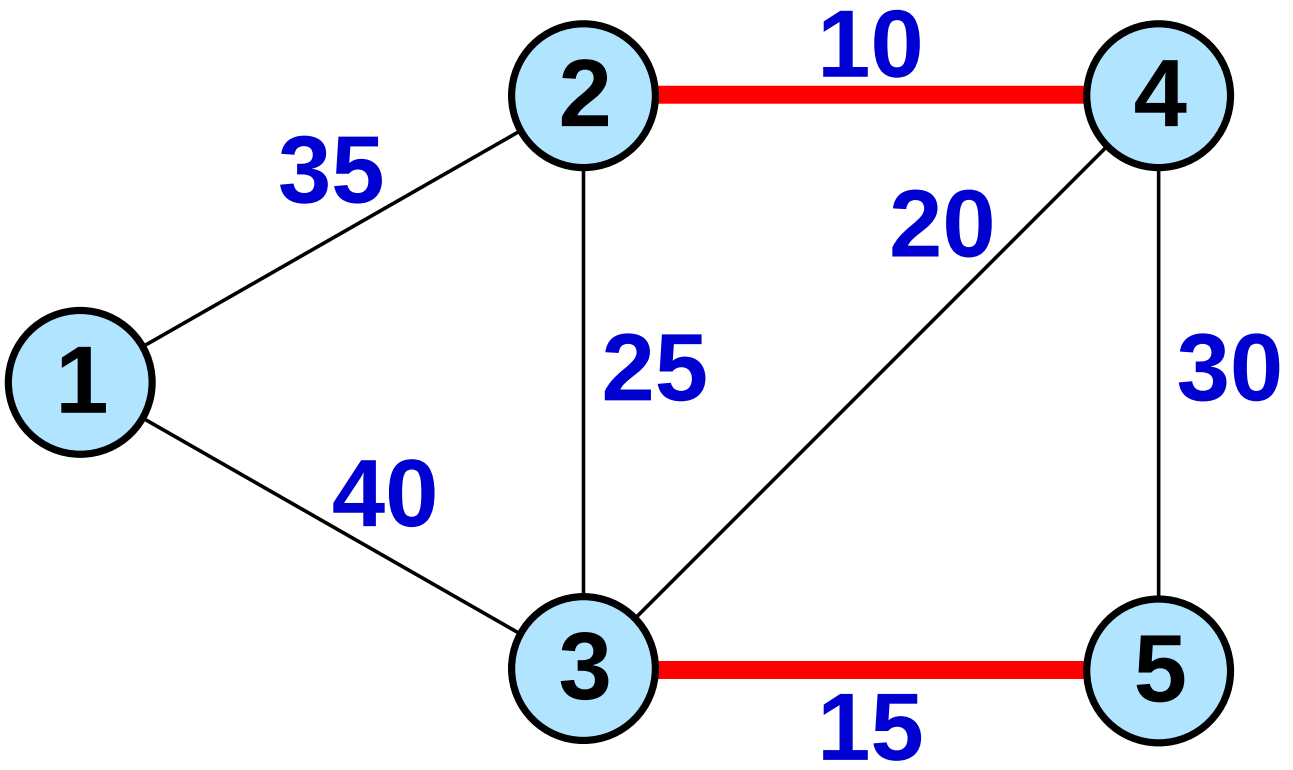
Step 1: consider link (2,4)



Add the link to the forest.

Kruskal's method example

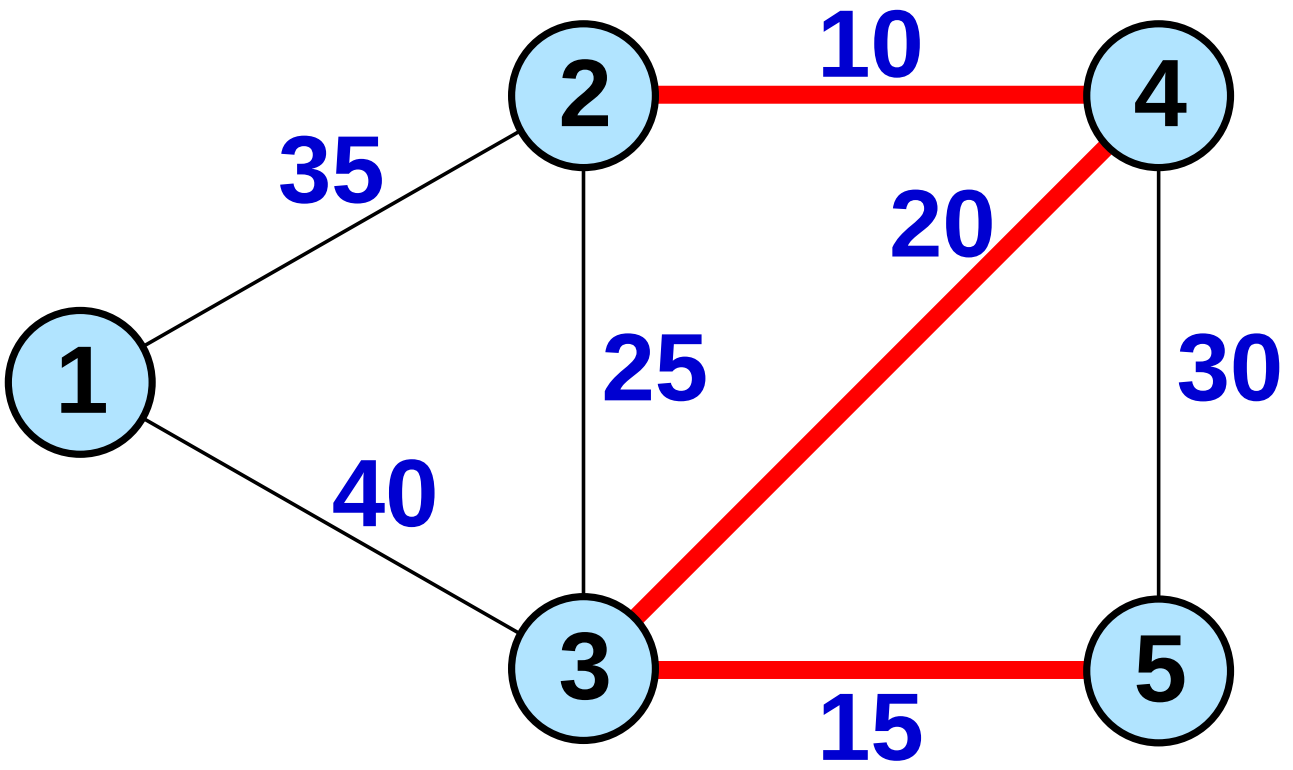
Step 2: consider link (3,5)



Add the link to the forest.

Kruskal's method example

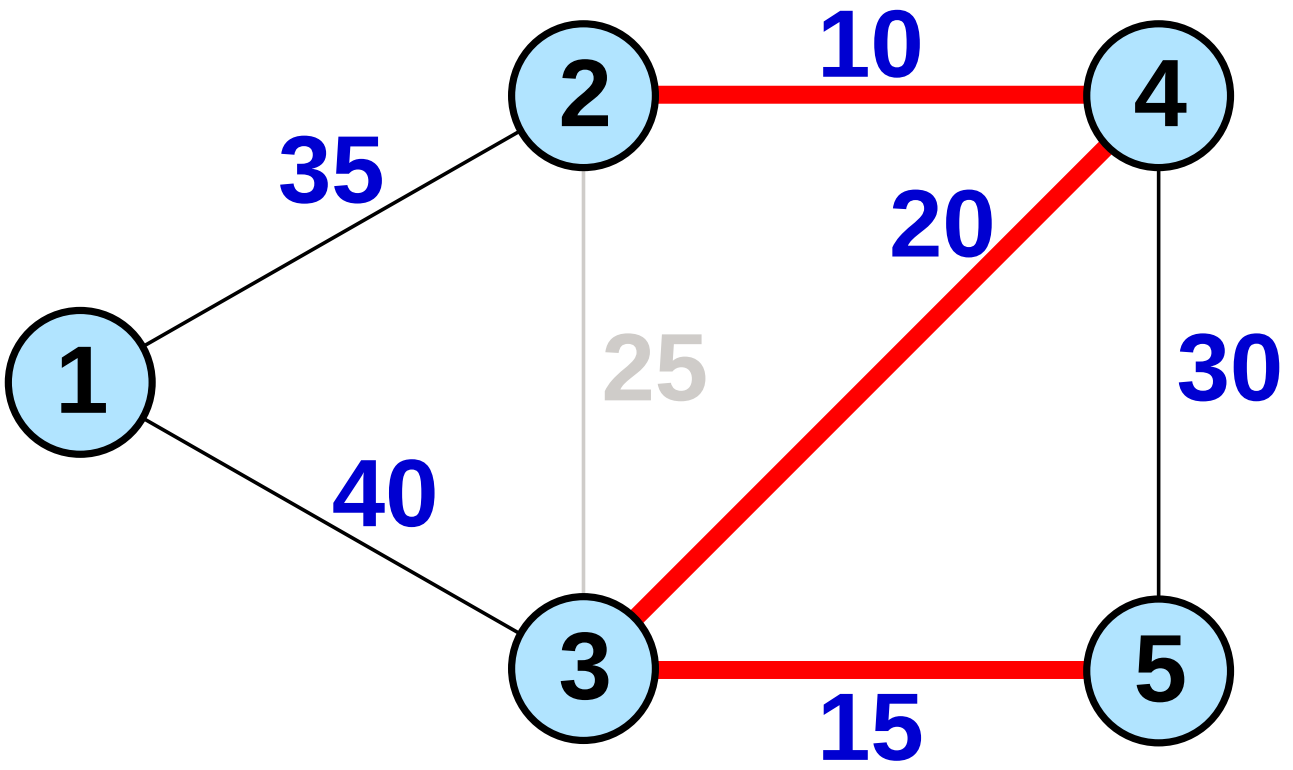
Step 3: consider link (3,4)



Add the link to the forest.

Kruskal's method example

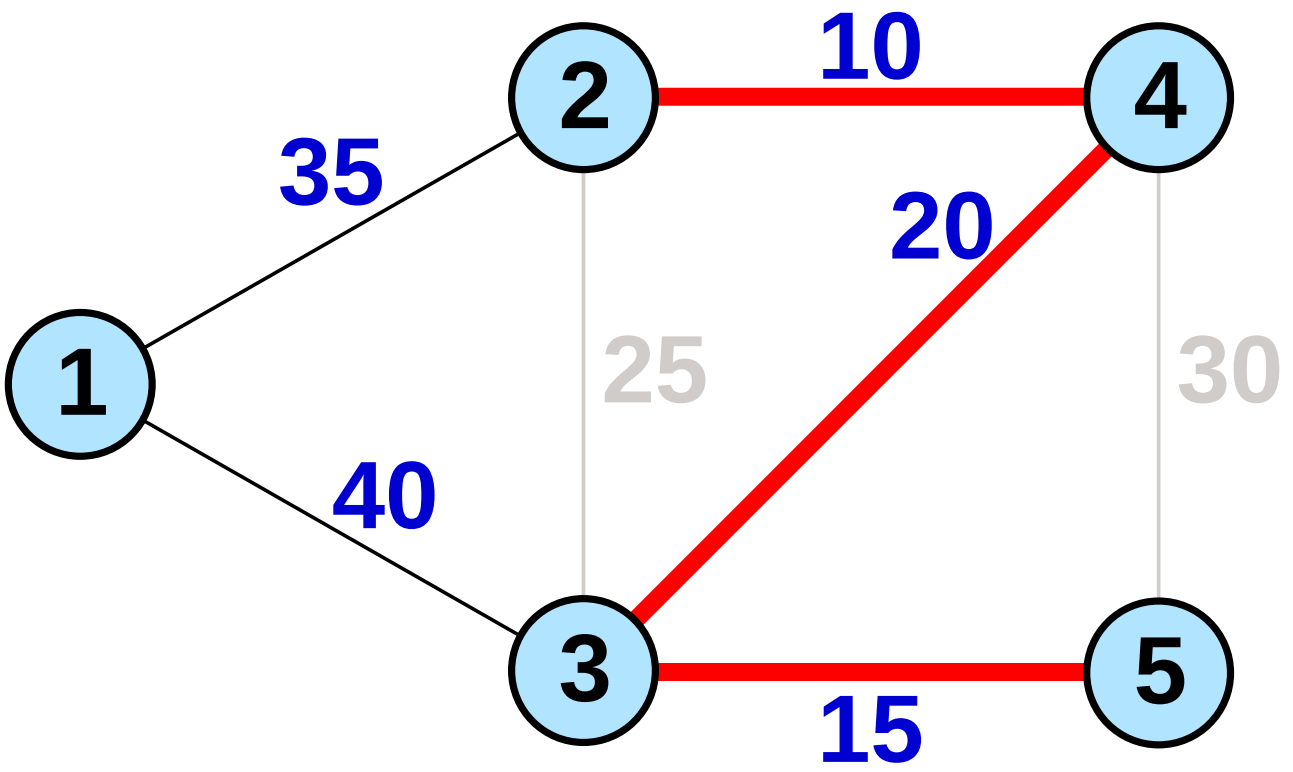
Step 4: consider link (2,3)



Discard the link as it would create a cycle

Kruskal's method example

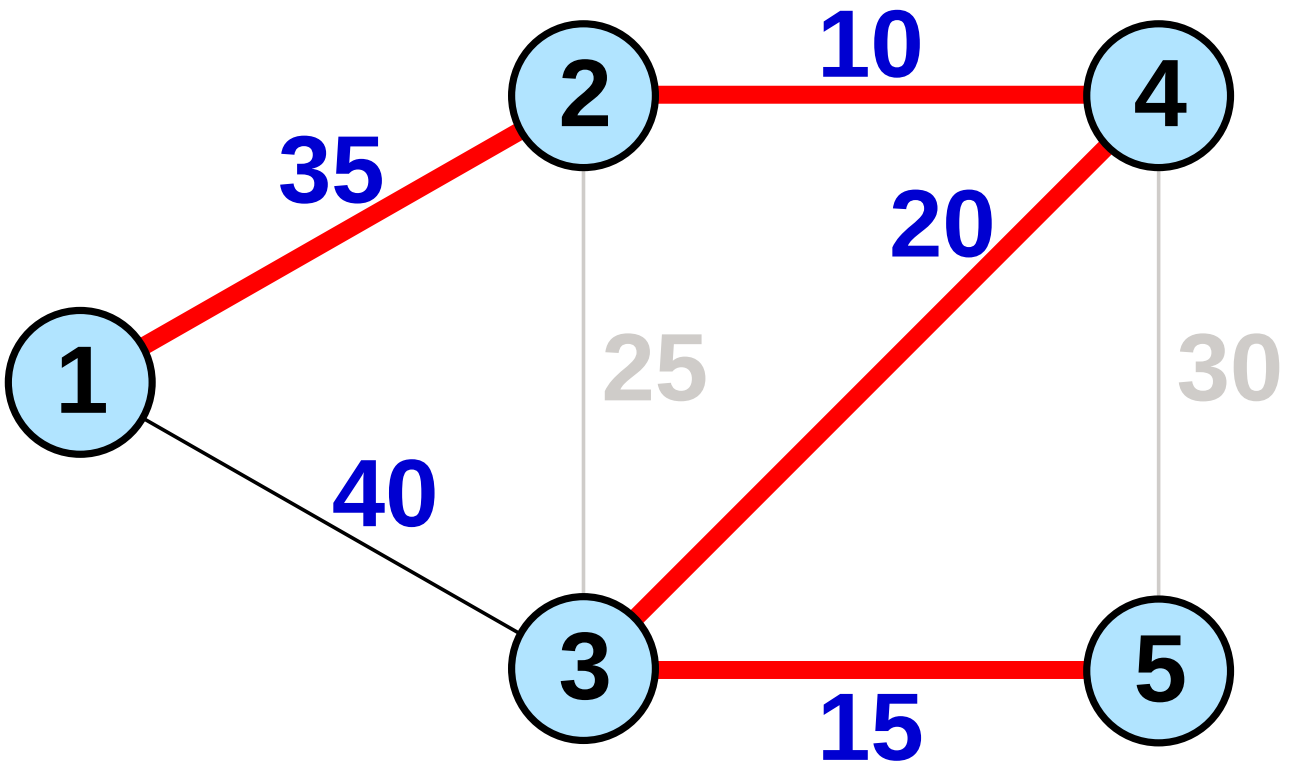
Step 5: consider link (4,5)



Discard the link as it would create a cycle

Kruskal's method example

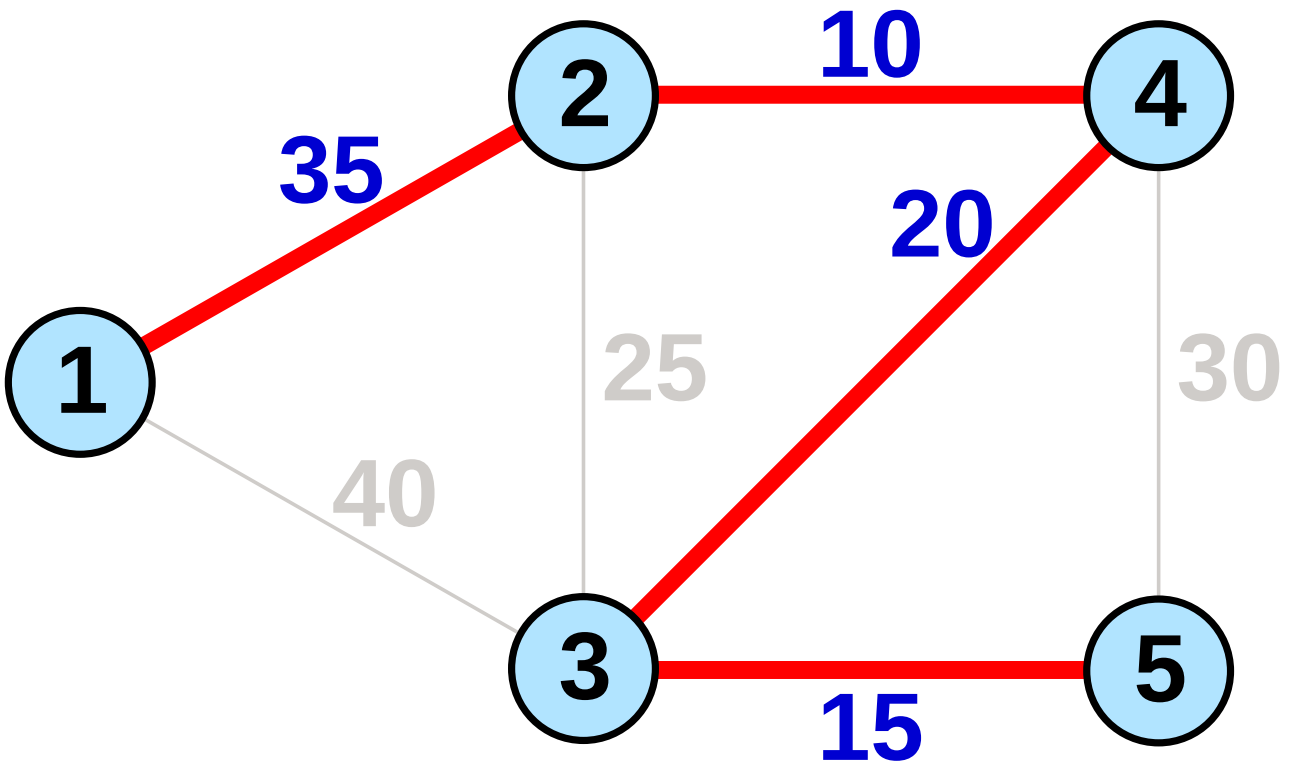
Step 6: consider link (1,2)



Add the link to the forest.
We have a spanning tree now, so we could stop here, and discard (1,3), but we will continue until the end.

Kruskal's method example

Step 7: consider link (1,3)



Discard the link as it would create a cycle

Kruskal's method proof

Sketch proof that Kruskal's method produces a MWST

- begin with a forest consisting of all nodes, separate
- at each stage, we have a forest (no cycles)
  - from prop.1, we can find such a forest
  - the algorithm itself prohibits new cycles forming
- subtrees in the forest are fragments of the MWST
  - we always add least weight links (without cycles)
  - new subtrees must be fragment of the MWSTs by prop.2
- final result must be the MWST

For full proofs see [2, Section 44, Part II], or [3, Thm 6.3]

Kruskal's method's complexity

- we start by sorting which takes $O(|E|\log|E|)$
  - for a completely connected network (the worst case) $O(|E|) = O(|N|^2)$
  - so $O(|E|\log|E|) = O(|E|\log|N|^2) = O(|E|\log|N|)$
- go through all of the edges in turn so $O(|E|)$ steps
- in each step, we need to test for cycles
  - for $e = (i, j)$ check whether nodes i and j are in the same connected sub-tree
  - varies with the method used
  - simple method is $O(|N|)$
  - careful method (see [3, Thm 6.4]) is $O(\log|N|)$
- Total complexity $O(|E|\log|N|)$

Prim's Method [4]

- Rather than add links with increasing weights, Prim's method fans out from a single arbitrary node.
- maintain a sub-tree (S, L)
  - $S \subset N$
  - $L \subset E$ such that L is a spanning tree on S
    - connects S
    - has no cycles
- at each step, add the "nearest neighbour" to S
  - add the cheapest link from S to $S \backslash N$
- using ideas from proposition 1 and 2

Prim's Method [4]

Input a connected network $G(N,E)$ with link weights $\beta_e \geq 0, \forall e \in E$

1. **Initialize:** $L = \phi,\ S = \{1\}$
2. **While** (S,L) not a spanning tree ($|L| < |N| - 1$)
   - take (i', j') such that

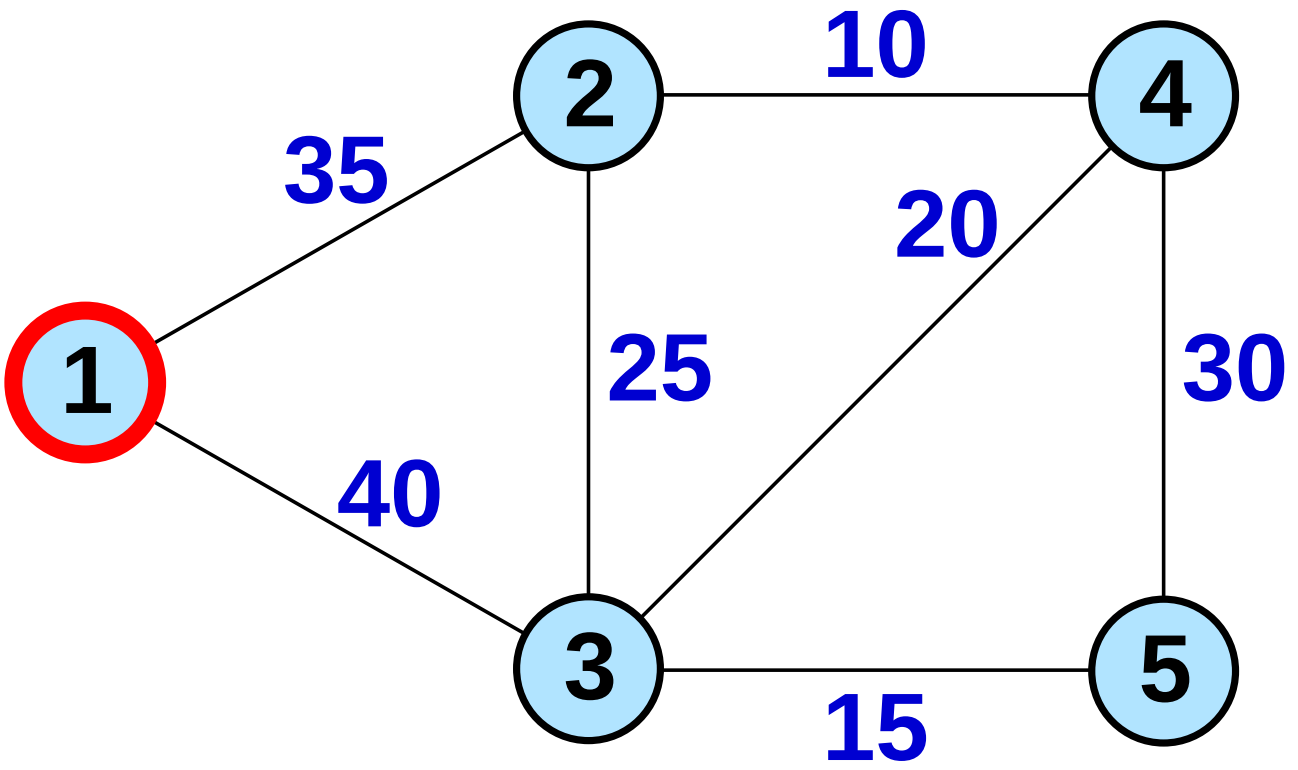
$$\beta_{i'j'} = \min\{\beta_{i,j} | (i,j) \in E, i \in S, j \notin S\}$$

   - add (i', j') to the list L and j' to S

Prim's method example

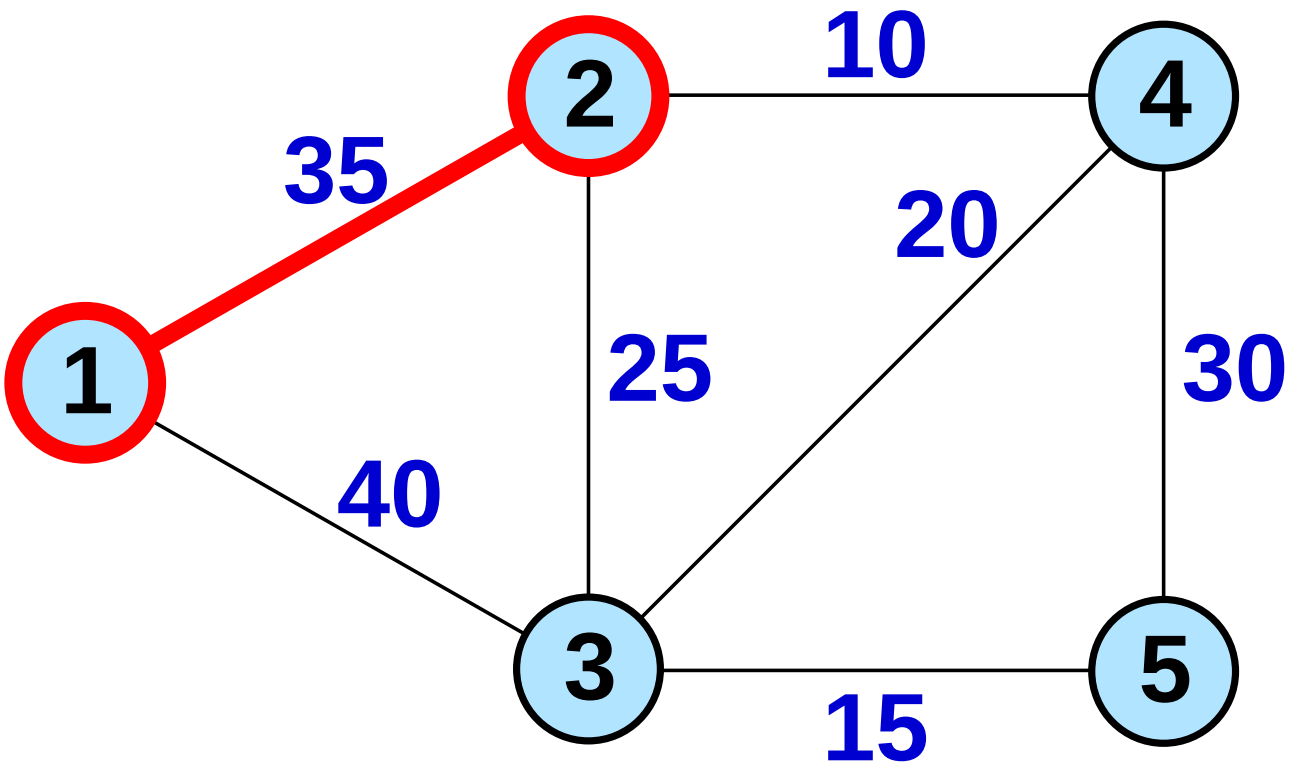
Input graph with link weights β_e (same as above)

Choose an arbitrary start node (we choose node 1)



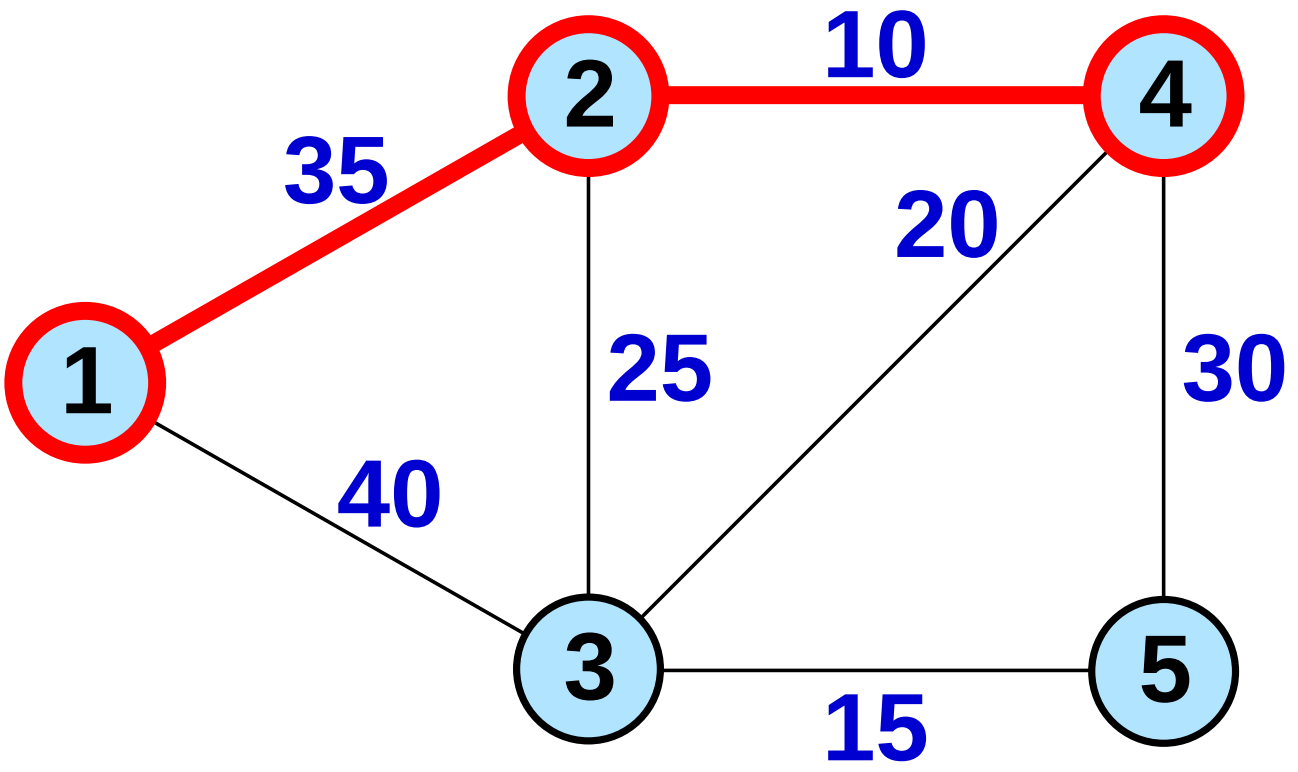
$S = \{1\}, \ L = \phi$

Prim's method example



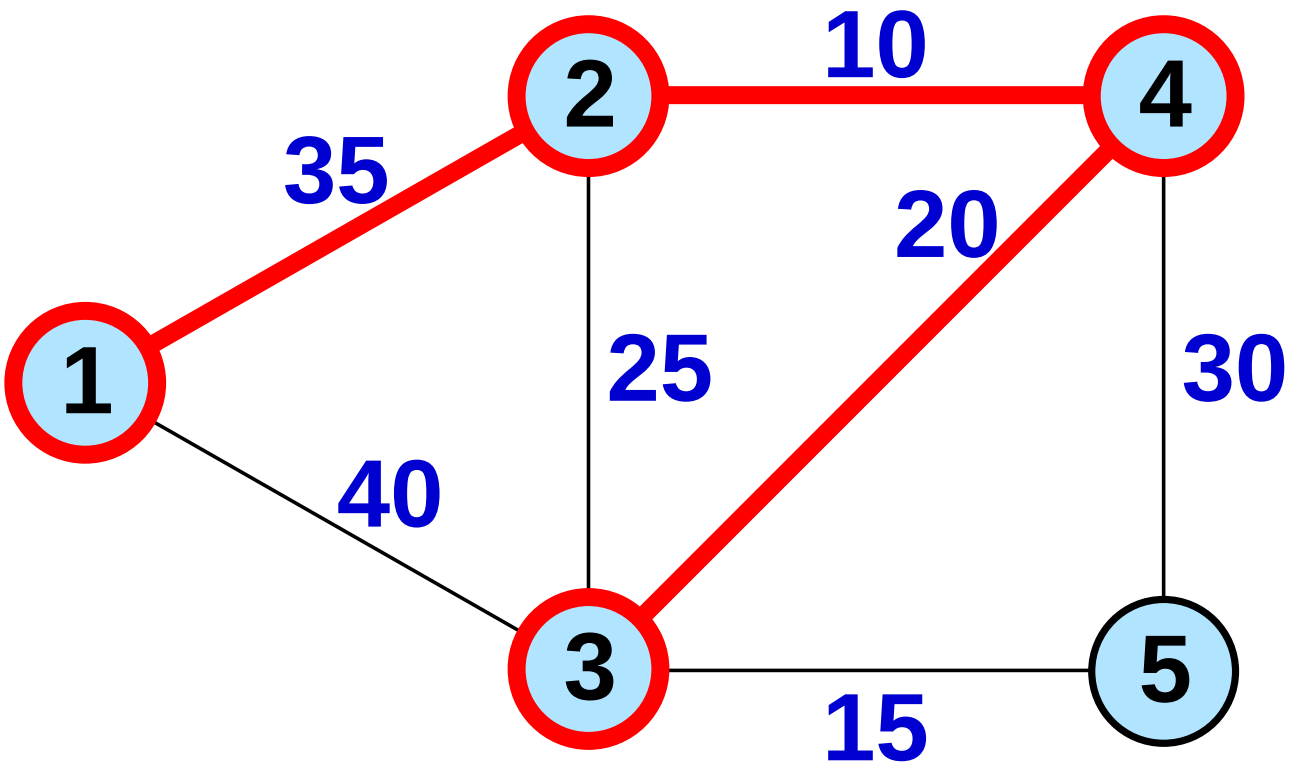
$S = \{1,2\},\ L = \{(1,2)\}$

Prim's method example



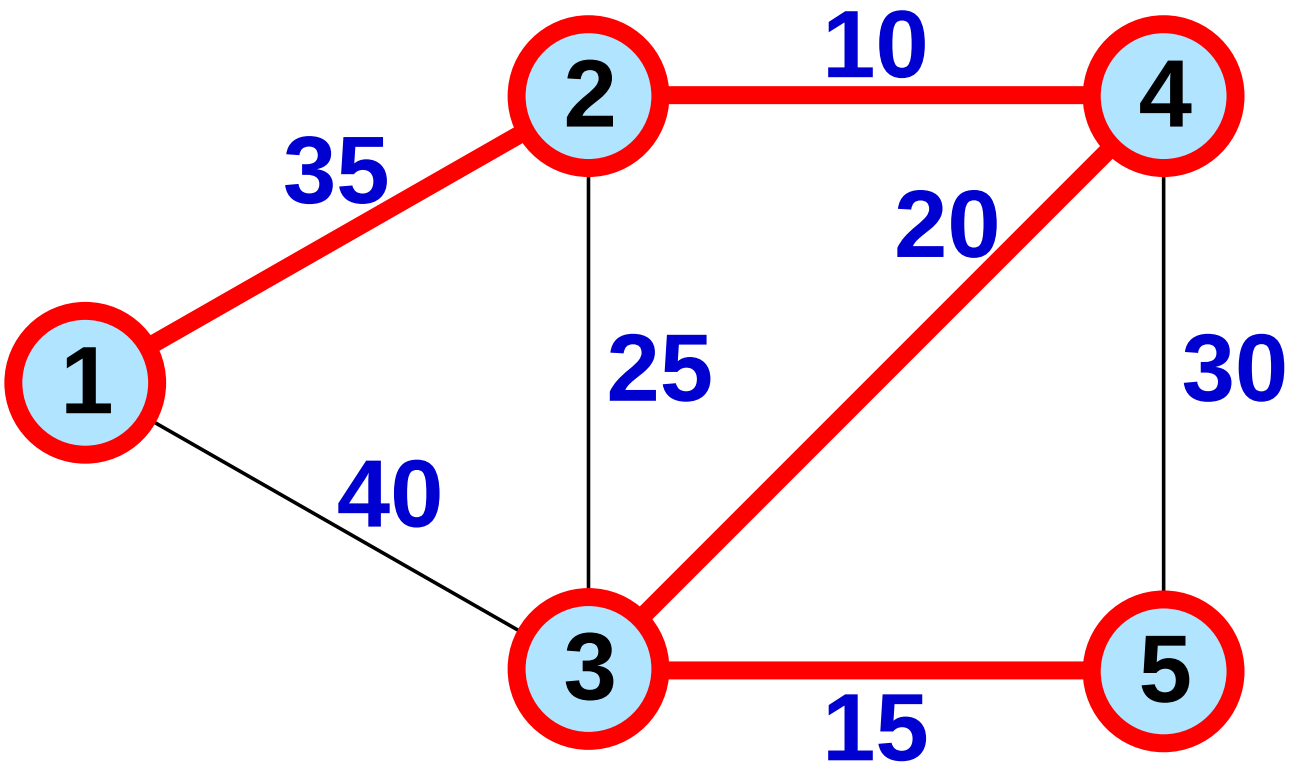
$S = \{1, 2, 4\},\ L = \{(1, 2), (2, 4)\}$

Prim's method example



$S = \{1, 2, 4, 3\}, \;\; L = \{(1,2), (2,4), (3,4)\}$

Prim's method example



$S = \{1, 2, 4, 3, 5\}, \;\; L = \{(1,2), (2,4), (3,4), (3,5)\}$

Prim's method details

- proof is straight from prop. 1 and prop. 2 again
- complexity of simple approach
  - need one step per node, so $O(|N|)$ steps
  - each step requires choice of min over the edges which takes time $O(|E|) = O(|N|)$ for a dense graph
  - total complexity is $O(|N|^2)$
- better approach, using a Fibonacci heap [3, p.121]
  - for a sparse graph this is $O(|E| + |N| \log |N|)$

Relationship to other algorithms

- **greedy** tree search algorithms are all similar
  - like Dijkstra, but unlike arbitrary greedy methods, these are all optimal
  - difference between Dijkstra and Prim
    - Dijkstra looks for next node to be closest to root
    - Prim looks for next node to be closest to current MWST
    - STP is really doing Dijkstra to get SPF tree
- A **tour** (minus a link) is a special case of a MWST
  - TSP results in a special case of a MWST
  - TSP is a cost minimization over a strictly smaller set
  - So in general the MWST cost is less than the TSP cost

Miscellany

- if each link has a distinct edge weight, there will be a unique MWST
  - in general the MWST is not unique
- The first algorithm for finding a minimum spanning tree was developed by Otakar Boruvka in 1926 [5].
- The fastest minimum spanning tree algorithm to date was developed by Bernard Chazelle, running time $O(|E|\alpha(|E|,|N|))$ where α is the classical functional inverse of an Ackermann function (effectively a constant here) [6].

References

[1] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," Proc. Amer. Math. Soc., vol. 7, pp. 48–50, 1956.

[2] A.Kaufmann, Graphs, Dynamic Programming, and Finite Games. Academic Press, 1967.

[3] B. Korte and J. Vygen, Combinatorial Optimization. Springer, 2000.

[4] R. C. Prim, "Shortest connection networks and some generalizations," Bell System Tech. J., vol. 36, pp. 1389–1401, 1957.

[5] O. Boruvka, "On minimum spanning tree problem," Discrete Mathematics, vol. 233, 2001. (translation by Jaroslav Nesetril, Eva Milkoá, Helena Nesetrilová).

[6] B. Chazelle, "A minimum spanning tree algorithm with inverse-ackermann type complexity," J. ACM, vol. 47, no. 6, pp. 1028–1047, 2000.