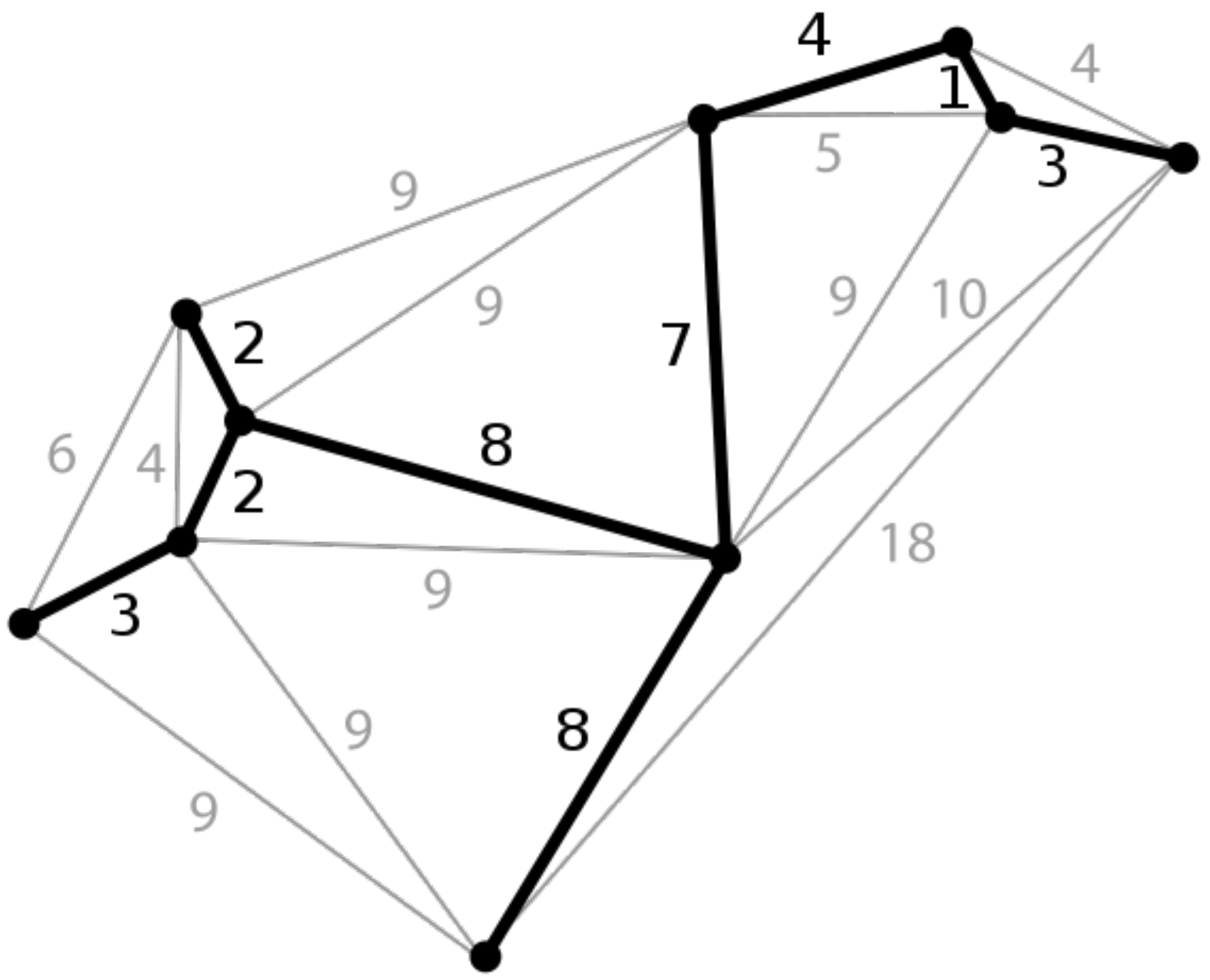
CS 106B
Lecture 25: Graphs II

Wednesday, May 30, 2018

Programming Abstractions
Spring 2018
Stanford University
Computer Science Department

Lecturer: Chris Gregg

reading:
Programming Abstractions in C++, Chapter 18



Today's Topics

- Logistics:
  - Final review session will be early next week.
  - If you need accommodations for the final exam, let Chris and Nick know now.
  - The Final will be using BlueBook — you'll need 3 hours of battery life, or find an outlet (we will try to provide enough outlets for those who need it)

- Real Graphs:
  - Internet routers and traceroute
- Topological Sort
- Minimum Spanning Trees
  - Kruskal's algorithm

Real Graphs!

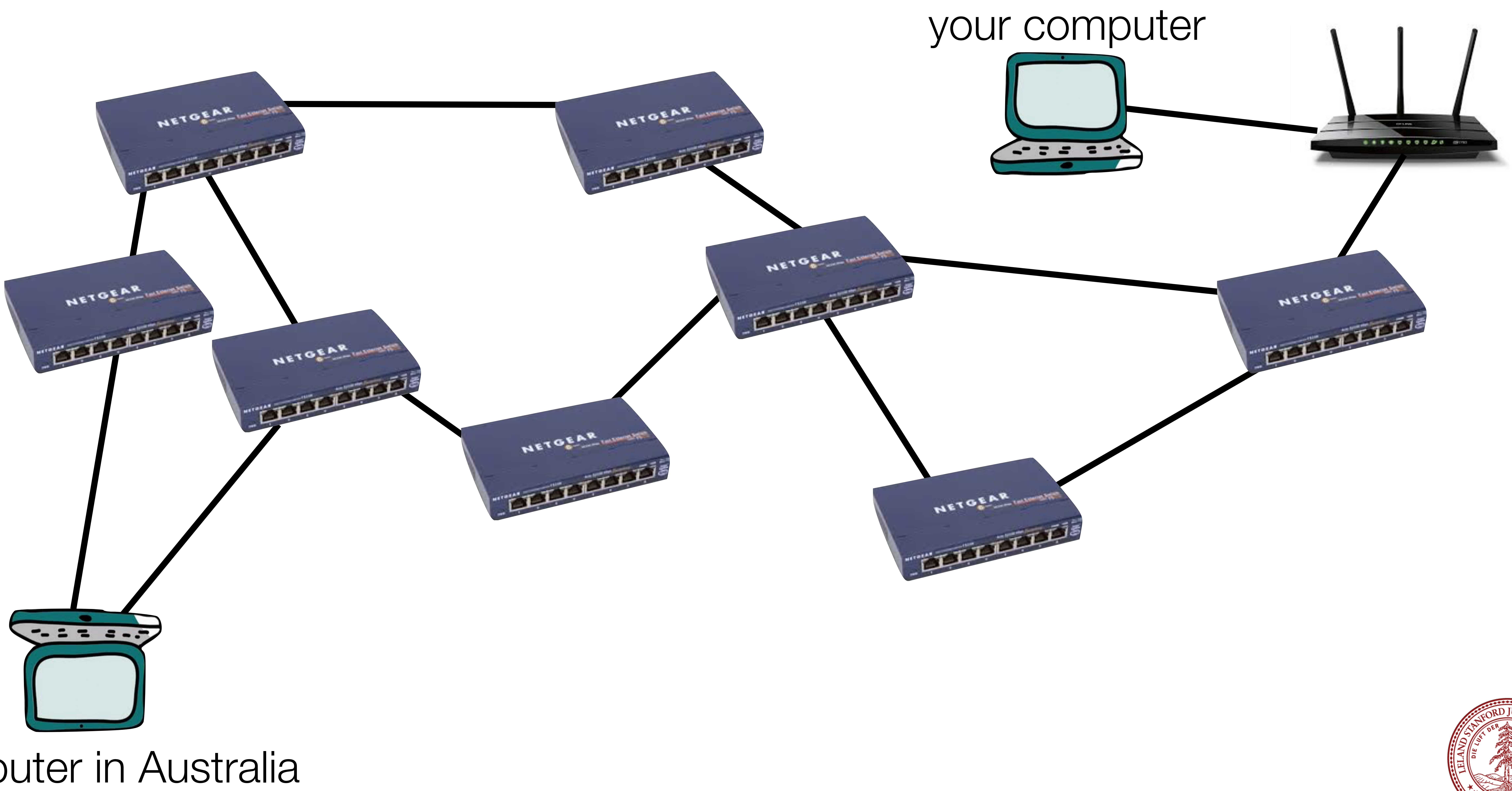
I received some feedback from last quarter that said,

> *“I would give more examples of how graphs are used, or, in general, give a wider variety of examples of applications to the methods being used. I think it would be interesting to give examples of how certain topics can be used in cross-sections of CS and other majors/departments.”*

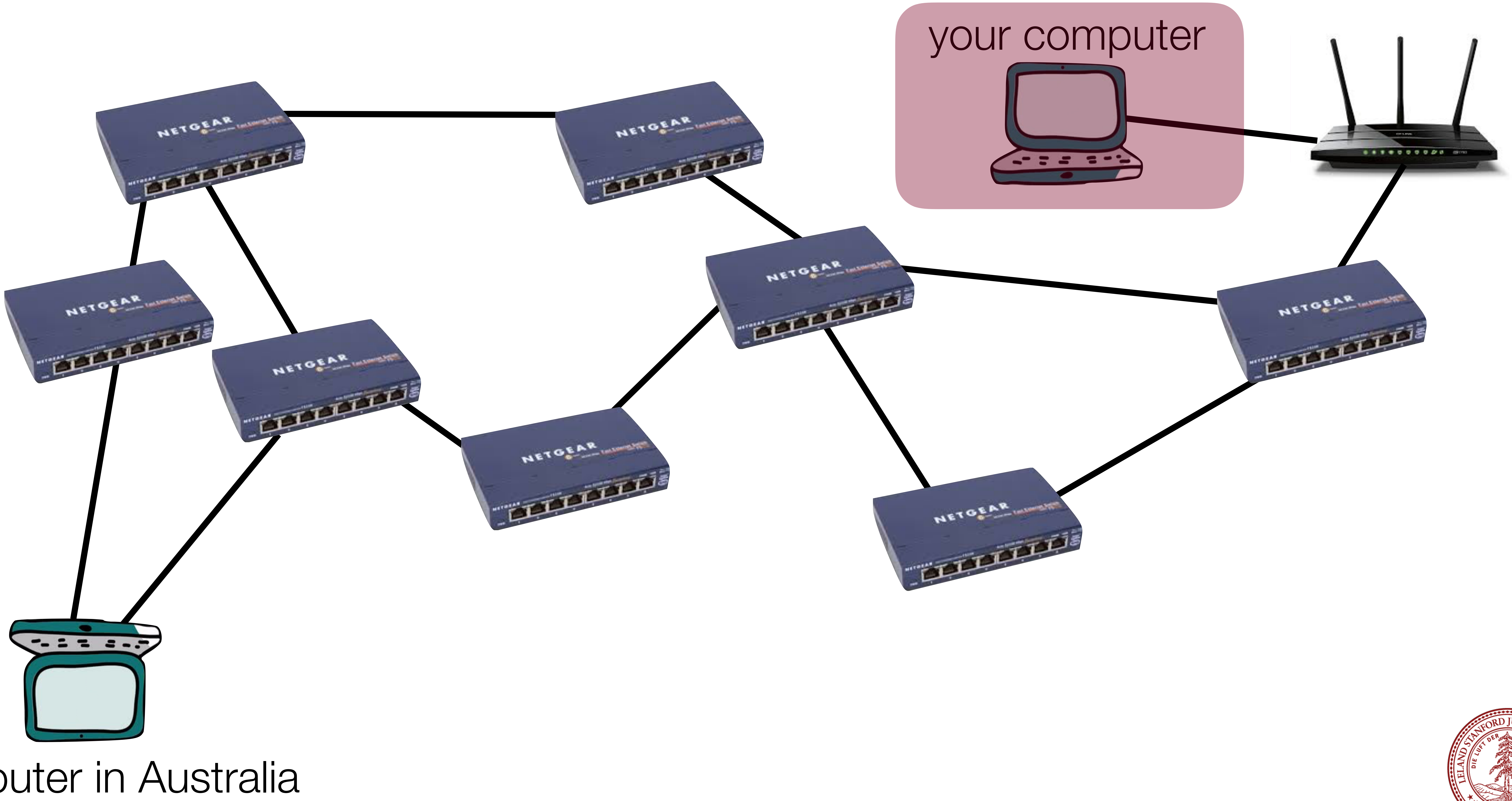
Let's dig a bit deeper into how the Internet is a real graph by analyzing internet routers, or:

How does a message get sent from your computer to another computer on the Internet, say in Australia?

The Internet: Computers connected through routers



The Internet: Computers connected through routers



The Internet: Let's simplify a bit

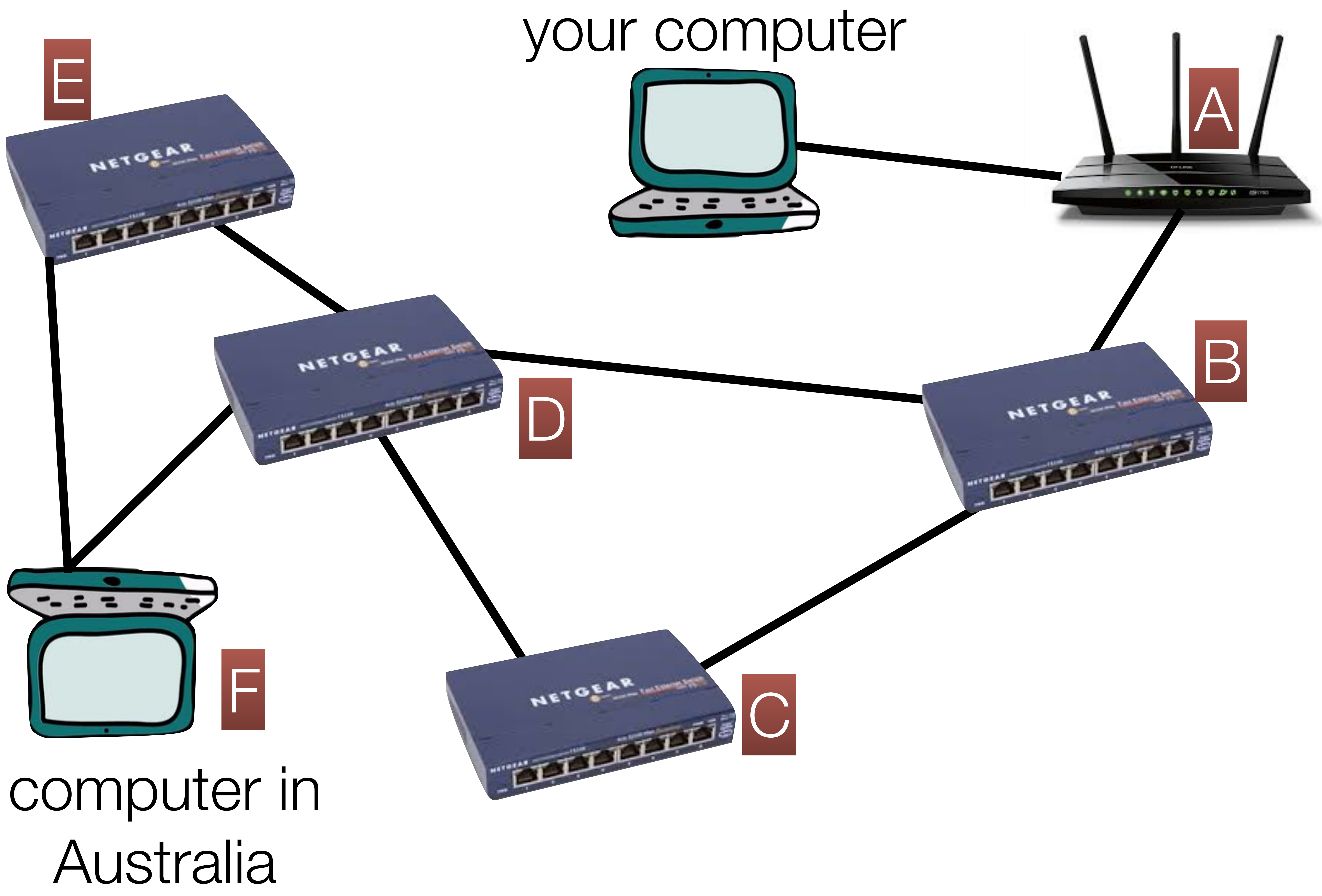
The destination computer has a name and an IP address, like this:

`www.engineering.unsw.edu.au`
`IP address: 149.171.158.109`

The first number denotes the "network address" and routers continually pass around information about how many "hops" they think it will take for them to get to all the networks.

E.g., for router `C`:

router	hops
A	2
B	1
C	-
D	1
E	2
F	2



The Internet: Let's simplify a bit

Each router knows its neighbors, and it has a copy of its neighbors' tables. So, `B` would have the following tables:

A

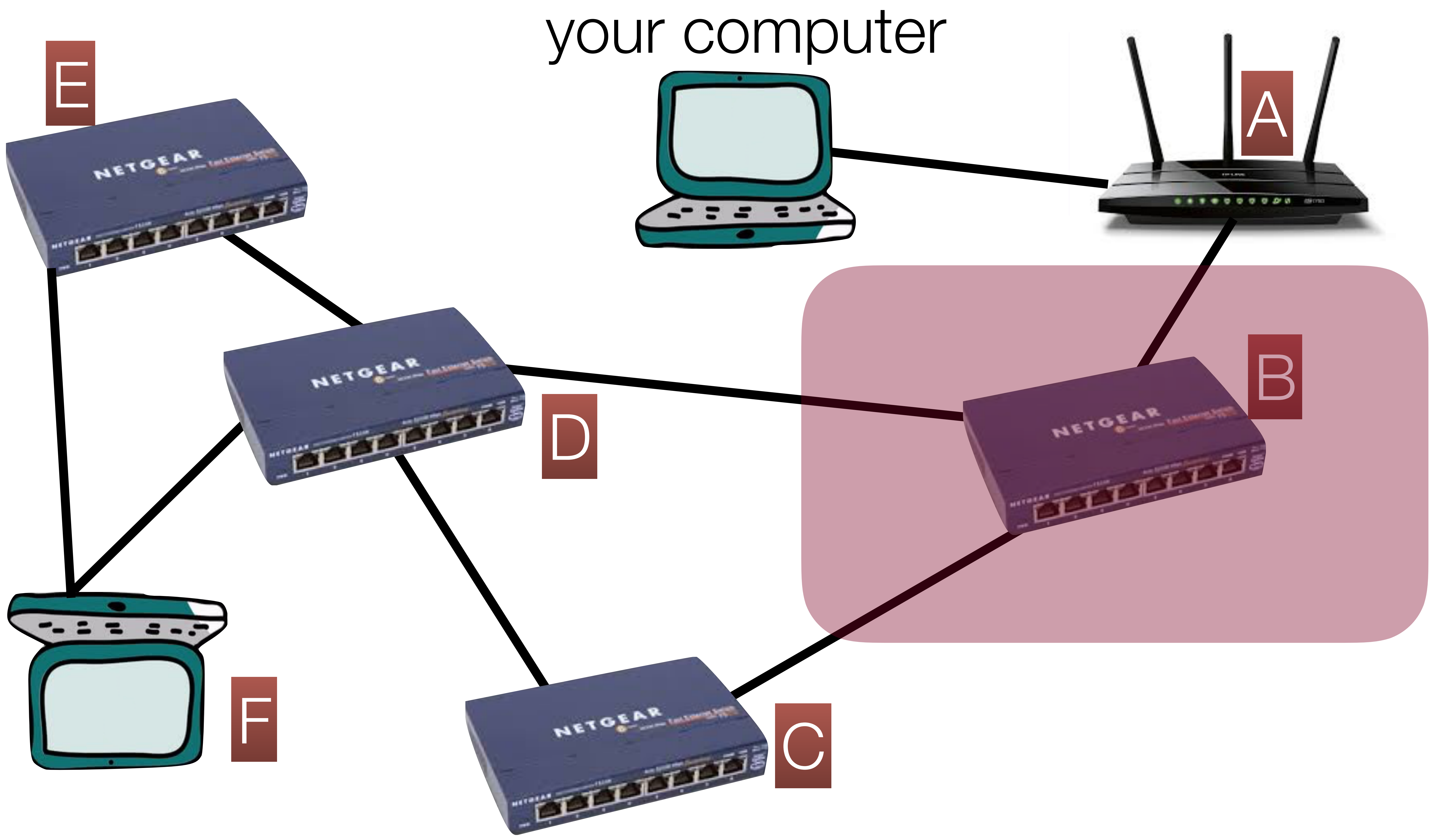
router	hops
A	-
B	1
C	3
D	2
E	3
F	3

C

router	hops
A	2
B	1
C	-
D	1
E	2
F	2

D

router	hops
A	2
B	1
C	1
D	-
E	1
F	1



The Internet: Let's simplify a bit

If B wants to connect to F, it connects through its neighbor that reports the shortest path to F. Which router would it choose?

A

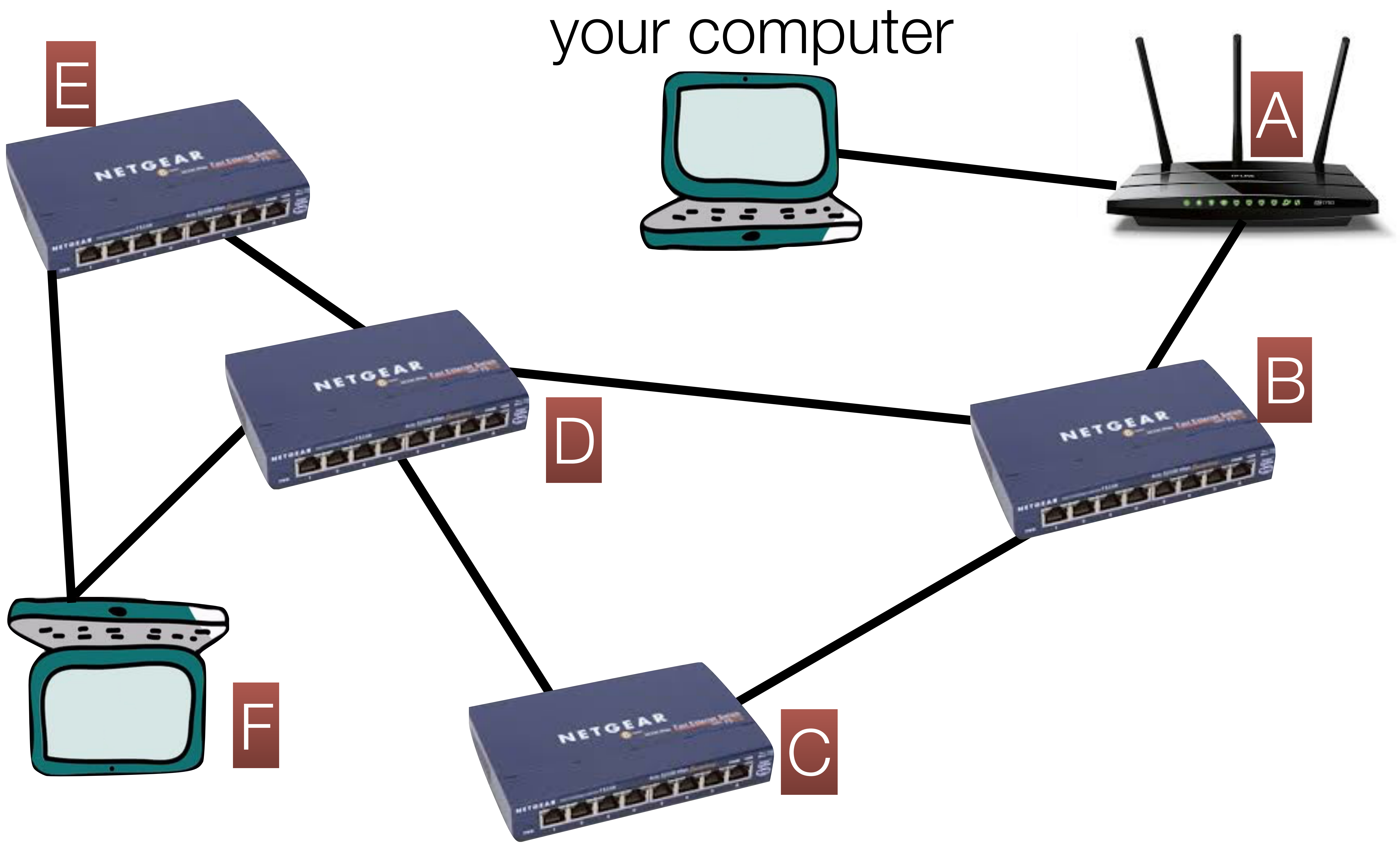
router	hops
A	-
B	1
C	3
D	2
E	3
F	3

C

router	hops
A	2
B	1
C	-
D	1
E	2
F	2

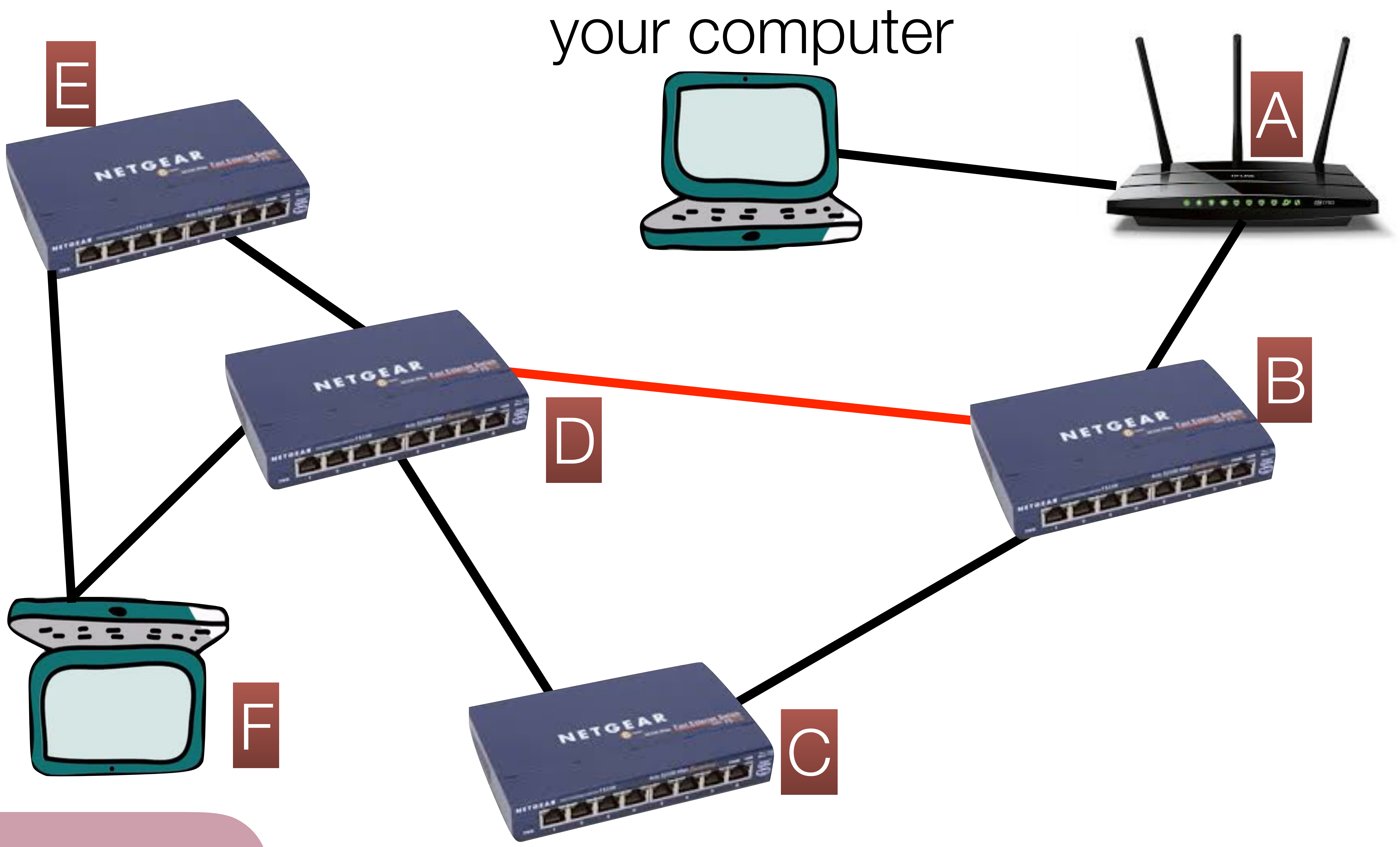
D

router	hops
A	2
B	1
C	1
D	-
E	1
F	1



The Internet: Let's simplify a bit

If B wants to connect to F, it connects through its neighbor that reports the shortest path to F. Which router would it choose? D.



A

router	hops
A	-
B	1
C	3
D	2
E	3
F	3

C

router	hops
A	2
B	1
C	-
D	1
E	2
F	2

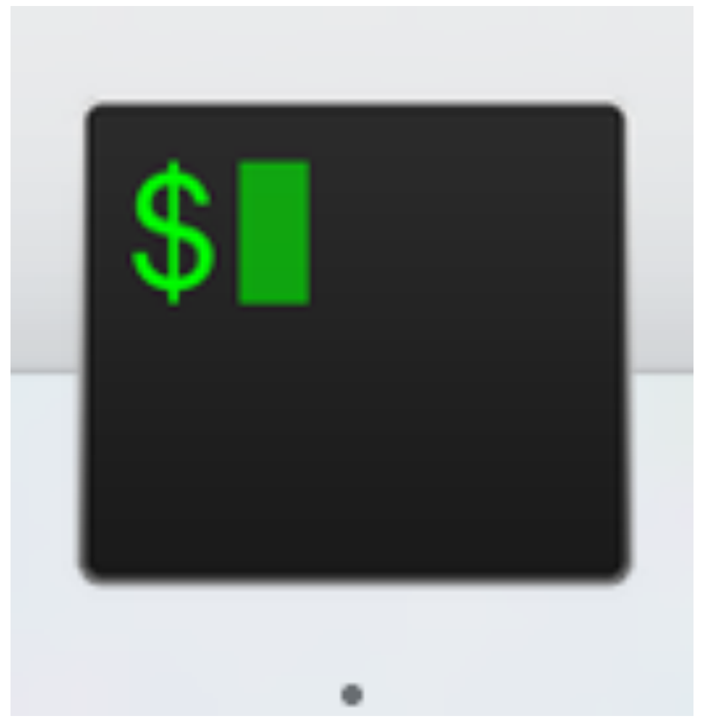
D

router	hops
A	2
B	1
C	1
D	-
E	1
F	1

Traceroute

We can use a program called "traceroute" to tell us the path between our computer and a different computer:

```
traceroute -I -e www.engineering.unsw.edu.au
```



Traceroute: Stanford Hops

```
traceroute -I -e www.engineering.unsw.edu.au
traceroute to www.engineering.unsw.edu.au (149.171.158.109), 64 hops max, 72 byte packets
 1  csmx-west-rtr.sunet (171.67.64.2)  5.965 ms  11.205 ms  14.180 ms
 2  gnat-1.sunet (172.24.70.11)  0.328 ms  0.312 ms  0.289 ms
 3  csmx-west-rtr.sunet (171.64.66.2)  15.702 ms  33.031 ms  10.003 ms
 4  dc-svl-rtr-vl8.sunet (171.64.255.204)  0.595 ms  0.552 ms  0.547 ms
 5  dc-svl-agg4--stanford-100ge.cenic.net (137.164.23.144)  1.618 ms  1.154 ms  1.672 ms
 6  hpr-svl-hpr2--svl-agg4-10ge.cenic.net (137.164.26.249)  1.056 ms  1.016 ms  0.929 ms
 7  aarnet-2-is-jmb-778.sttlwa.pacificwave.net (207.231.245.4)  17.803 ms  17.714 ms  17.978 ms
 8  et-2-0-0.pe1.a.hnl.aarnet.net.au (113.197.15.200)  69.984 ms  69.888 ms  70.004 ms
 9  et-2-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.98)  162.935 ms  163.033 ms  162.977 ms
10  et-3-3-0.pe1.brwy.nsw.aarnet.net.au (113.197.15.148)  163.819 ms  163.083 ms  163.111 ms
11  138.44.5.1 (138.44.5.1)  163.124 ms  163.236 ms  163.254 ms
12  libcr1-te-1-5.gw.unsw.edu.au (149.171.255.102)  163.448 ms  163.355 ms  163.307 ms
13  libdcdnex1-po-1.gw.unsw.edu.au (149.171.255.174)  163.544 ms  163.250 ms  163.344 ms
14  srdh4it2r26blfx1-ext.gw.unsw.edu.au (129.94.0.31)  164.605 ms  164.288 ms  164.461 ms
15  bfw1-ae-1-3053.gw.unsw.edu.au (129.94.254.76)  164.065 ms  164.066 ms  164.267 ms
16  engplws008.eng.unsw.edu.au (149.171.158.109)  164.326 ms  164.538 ms  164.462 ms
```

Traceroute: CENIC

```
traceroute -I -e www.engineering.unsw.edu.au
traceroute to www.engineering.unsw.edu.au (149.171.158.109), 64 hops max, 72 byte packets
 1  csmx-west-rtr.sunet (171.67.64.2)  5.965 ms  11.205 ms  14.180 ms
 2  gnat-1.sunet (172.24.70.11)  0.328 ms  0.312 ms  0.289 ms
 3  csmx-west-rtr.sunet (171.64.66.2)  15.702 ms  33.031 ms  10.003 ms
 4  dc-svl-rtr-vl8.sunet (171.64.255.204)  0.595 ms  0.552 ms  0.547 ms
 5  dc-svl-agg4--stanford-100ge.cenic.net (137.164.23.144)  1.618 ms  1.154 ms  1.672 ms
 6  hpr-svl-hpr2--svl-agg4-10ge.cenic.net (137.164.26.249)  1.056 ms  1.016 ms  0.929 ms
 7  aarnet-2-is-jmb-778.sttlwa.pacificwave.net (207.231.245.4)  17.803 ms  17.714 ms  17.978 ms
 8  et-2-0-0.pe1.a.hnl.aarnet.net.au (113.197.15.200)  69.984 ms  69.888 ms  70.004 ms
 9  et-2-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.98)  162.935 ms  163.033 ms  162.977 ms
10  et-3-3-0.pe1.brwy.nsw.aarnet.net.au (113.197.15.148)  163.819 ms  163.083 ms  163.111 ms
11  138.44.5.1 (138.44.5.1)  163.124 ms  163.236 ms  163.254 ms
12  libcr1-te-1-5.gw.unsw.edu.au (149.171.255.102)  163.448 ms  163.355 ms  163.307 ms
13  libdcdnex1-po-1.gw.unsw.edu.au (149.171.255.174)  163.544 ms  163.250 ms  163.344 ms
14  srdh4it2r26blfx1-ext.gw.unsw.edu.au (129.94.0.31)  164.605 ms  164.288 ms  164.461 ms
15  bfw1-ae-1-3053.gw.unsw.edu.au (129.94.254.76)  164.065 ms  164.066 ms  164.267 ms
16  engplws008.eng.unsw.edu.au (149.171.158.109)  164.326 ms  164.538 ms  164.462 ms
```

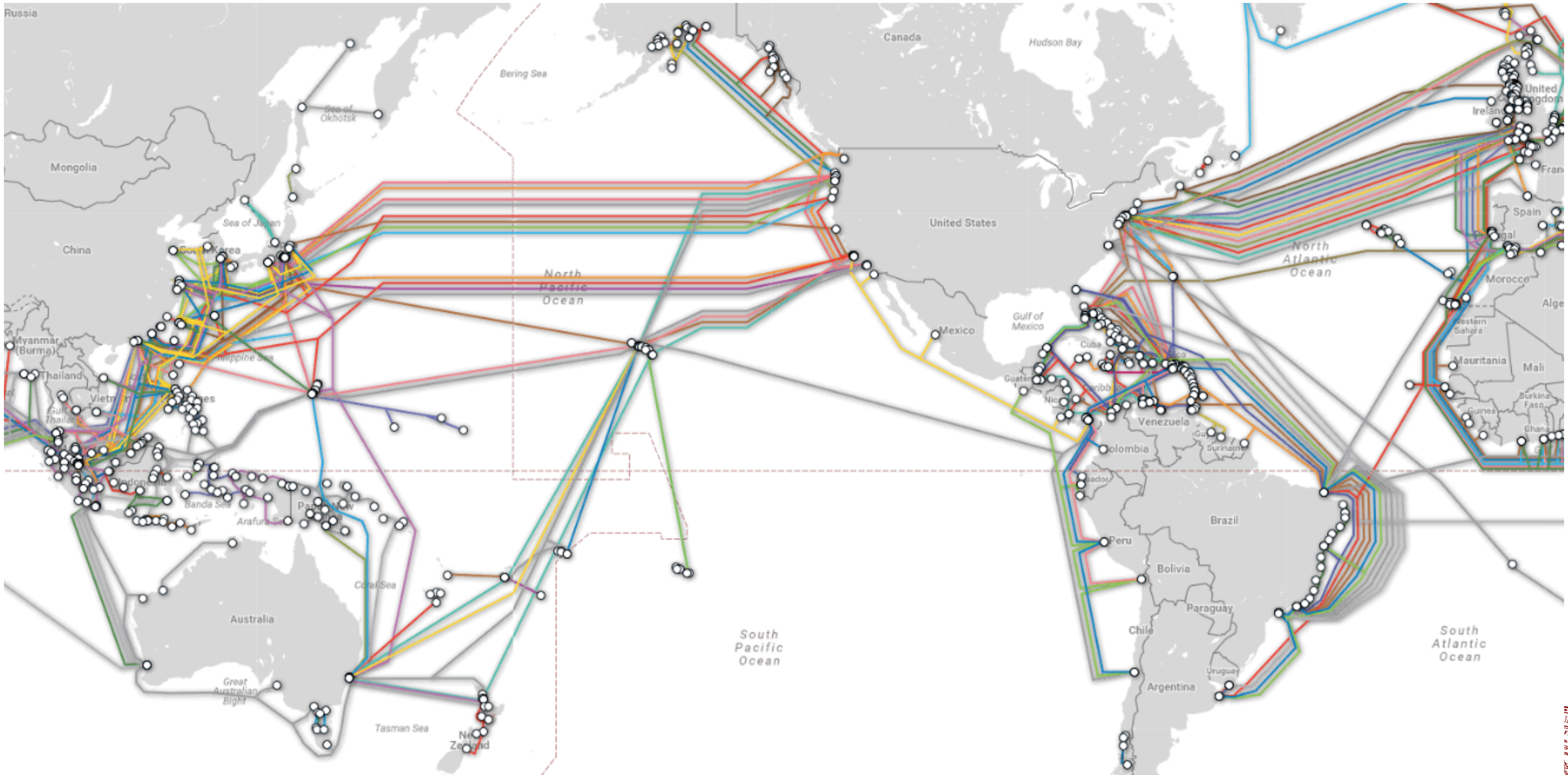
The **Corporation for Education Network Initiatives in California** (**CENIC**) is a nonprofit corporation formed in 1996 to provide high-performance, high-bandwidth networking services to California universities and research institutions (source: Wikipedia)

Traceroute: Pacificwave (Seattle)

```
traceroute -I -e www.engineering.unsw.edu.au
traceroute to www.engineering.unsw.edu.au (149.171.158.109), 64 hops max, 72 byte packets
 1  csmx-west-rtr.sunet (171.67.64.2)  5.965 ms  11.205 ms  14.180 ms
 2  gnat-1.sunet (172.24.70.11)  0.328 ms  0.312 ms  0.289 ms
 3  csmx-west-rtr.sunet (171.64.66.2)  15.702 ms  33.031 ms  10.003 ms
 4  dc-svl-rtr-vl8.sunet (171.64.255.204)  0.595 ms  0.552 ms  0.547 ms
 5  dc-svl-agg4--stanford-100ge.cenic.net (137.164.23.144)  1.618 ms  1.154 ms  1.672 ms
 6  hpr-svl-hpr2--svl-agg4-10ge.cenic.net (137.164.26.249)  1.056 ms  1.016 ms  0.929 ms
 7  aarnet-2-is-jmb-778.sttlwa.pacificwave.net (207.231.245.4)  17.803 ms  17.714 ms  17.978 ms
 8  et-2-0-0.pe1.a.hnl.aarnet.net.au (113.197.15.200)  69.984 ms  69.888 ms  70.004 ms
 9  et-2-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.98)  162.935 ms  163.033 ms  162.977 ms
10  et-3-3-0.pe1.brwy.nsw.aarnet.net.au (113.197.15.148)  163.819 ms  163.083 ms  163.111 ms
11  [illegible]8.44.5.1)  163.124 ms  163.236 ms  163.254 ms
12  [illegible]gw.unsw.edu.au (149.171.255.102)  163.448 ms  163.355 ms  163.307 ms
13  [illegible]-1.gw.unsw.edu.au (149.171.255.174)  163.544 ms  163.250 ms  163.344 ms
14  [illegible]fx1-ext.gw.unsw.edu.au (129.94.0.31)  164.605 ms  164.288 ms  164.461 ms
15  bfw1-ae-1-30[illegible].gw.unsw.edu.au (129.94.254.76)  164.065 ms  164.066 ms  164.267 ms
16  [illegible].unsw.edu.au (149.171.158.109)  164.326 ms  164.538 ms  164.462 ms
```

Pass internet traffic directly with other major national and international networks, including U.S. federal agencies and many Pacific Rim R&E networks (source: http://www.pnwgp.net/services/pacific-wave-peering-exchange/)

Traceroute: Oregon to Australia - underwater!



http://www.submarinecablemap.com

Traceroute: Australia

```
traceroute -I -e www.engineering.unsw.edu.au
traceroute to www.engineering.unsw.edu.au (149.171.158.109), 64 hops max, 72 byte packets
 1  csmx-west-rtr.sunet (171.67.64.2)  5.965 ms  11.205 ms  14.180 ms
 2  gnat-1.sunet (172.24.70.11)  0.328 ms  0.312 ms  0.289 ms
 3  csmx-west-rtr.sunet (171.64.66.2)  15.702 ms  33.031 ms  10.003 ms
 4  dc-svl-rtr-vl8.sunet (171.64.255.204)  0.595 ms  0.552 ms  0.547 ms
 5  dc-svl-agg4--stanford-100ge.cenic.net (137.164.23.144)  1.618 ms  1.154 ms  1.672 ms
 6  hpr-svl-hpr2--svl-agg4-10ge.cenic.net (137.164.26.249)  1.056 ms  1.016 ms  0.929 ms
 7  aarnet-2-is-jmb-778.sttlwa.pacificwave.net (207.231.245.4)  17.803 ms  17.714 ms  17.978 ms
 8  et-2-0-0.pe1.a.hnl.aarnet.net.au (113.197.15.200)  69.984 ms  69.888 ms  70.004 ms
 9  et-2-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.98)  162.935 ms  163.033 ms  162.977 ms
10  et-3-3-0.pe1.brwy.nsw.aarnet.net.au (113.197.15.148)  163.819 ms  163.083 ms  163.111 ms
11  138.44.5.1 (138.44.5.1)  163.124 ms  163.236 ms  163.254 ms
12  libcr1-te-1-5.gw.unsw.edu.au (149.171.255.102)  163.448 ms  163.355 ms  163.307 ms
13  libdcdnex1-po-1.gw.unsw.edu.au (149.171.255.174)  163.544 ms  163.250 ms  163.344 ms
14  srdh4it2r26blfx1-ext.gw.unsw.edu.au (129.94.0.31)  164.605 ms  164.288 ms  164.461 ms
15  bfw1-ae-1-3053.gw.unsw.edu.au (129.94.254.76)  164.065 ms  164.066 ms  164.267 ms
16  engplws008.eng.unsw.edu.au (149.171.158.109)  164.326 ms  164.538 ms  164.462 ms
```

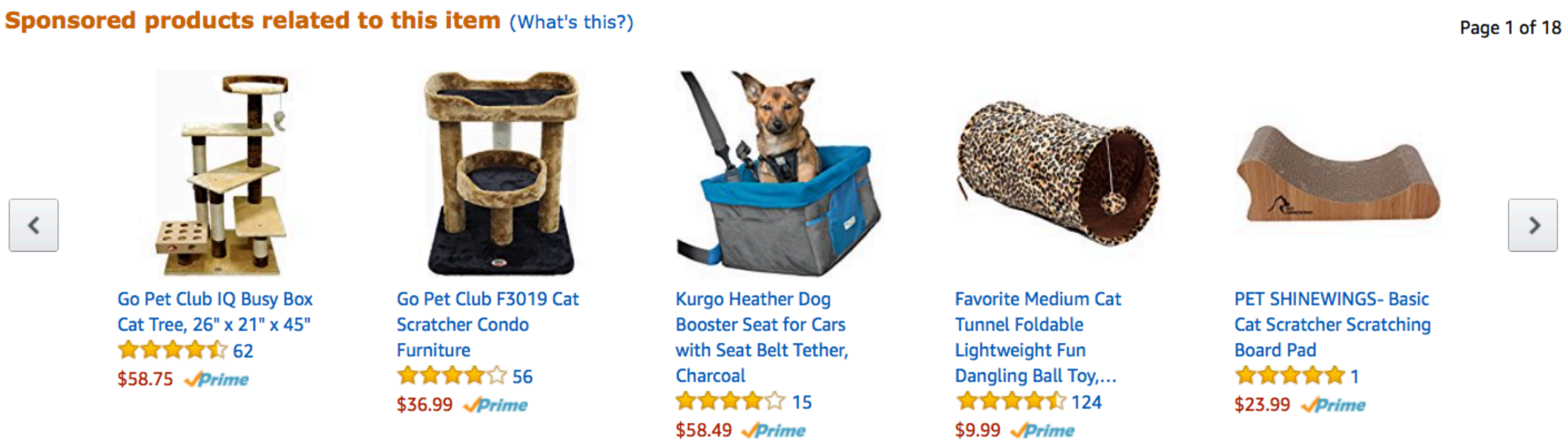
Traceroute: University of New South Wales

```
traceroute -I -e www.engineering.unsw.edu.au
traceroute to www.engineering.unsw.edu.au (149.171.158.109), 64 hops max, 72 byte packets
 1  csmx-west-rtr.sunet (171.67.64.2)  5.965 ms  11.205 ms  14.180 ms
 2  gnat-1.sunet (172.24.70.11)  0.328 ms  0.312 ms  0.289 ms
 3  csmx-west-rtr.sunet (171.64.66.2)  15.702 ms  33.031 ms  10.003 ms
 4  dc-svl-rtr-vl8.sunet (171.64.255.204)  0.595 ms  0.552 ms  0.547 ms
 5  dc-svl-agg4--stanford-100ge.cenic.net (137.164.23.144)  1.618 ms  1.154 ms  1.672 ms
 6  hpr-svl-hpr2--svl-agg4-10ge.cenic.net (137.164.26.249)  1.056 ms  1.016 ms  0.929 ms
 7  aarnet-2-is-jmb-778.sttlwa.pacificwave.net (207.231.245.4)  17.803 ms  17.714 ms  17.978 ms
 8  et-2-0-0.pe1.a.hnl.aarnet.net.au (113.197.15.200)  69.984 ms  69.888 ms  70.004 ms
 9  et-2-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.98)  162.935 ms  163.033 ms  162.977 ms
10  et-3-3-0.pe1.brwy.nsw.aarnet.net.au (113.197.15.148)  163.819 ms  163.083 ms  163.111 ms
11  138.44.5.1 (138.44.5.1)  163.124 ms  163.236 ms  163.254 ms
12  libcr1-te-1-5.gw.unsw.edu.au (149.171.255.102)  163.448 ms  163.355 ms  163.307 ms
13  libdcdnex1-po-1.gw.unsw.edu.au (149.171.255.174)  163.544 ms  163.250 ms  163.344 ms
14  srdh4it2r26blfx1-ext.gw.unsw.edu.au (129.94.0.31)  164.605 ms  164.288 ms  164.461 ms
15  bfw1-ae-1-3053.gw.unsw.edu.au (129.94.254.76)  164.065 ms  164.066 ms  164.267 ms
16  engplws008.eng.unsw.edu.au (149.171.158.109)  164.326 ms  164.538 ms  164.462 ms
```

161 milliseconds to get to the final computer

Other Real Life Uses for Graphs

- Amazon.com -- Product relationships are graph-based



Sponsored products related to this item (What's this?)

Page 1 of 18

Go Pet Club IQ Busy Box Cat Tree, 26" x 21" x 45"
★★★★½ 62
$58.75 Prime

Go Pet Club F3019 Cat Scratcher Condo Furniture
★★★★☆ 56
$36.99 Prime

Kurgo Heather Dog Booster Seat for Cars with Seat Belt Tether, Charcoal
★★★★☆ 15
$58.49 Prime

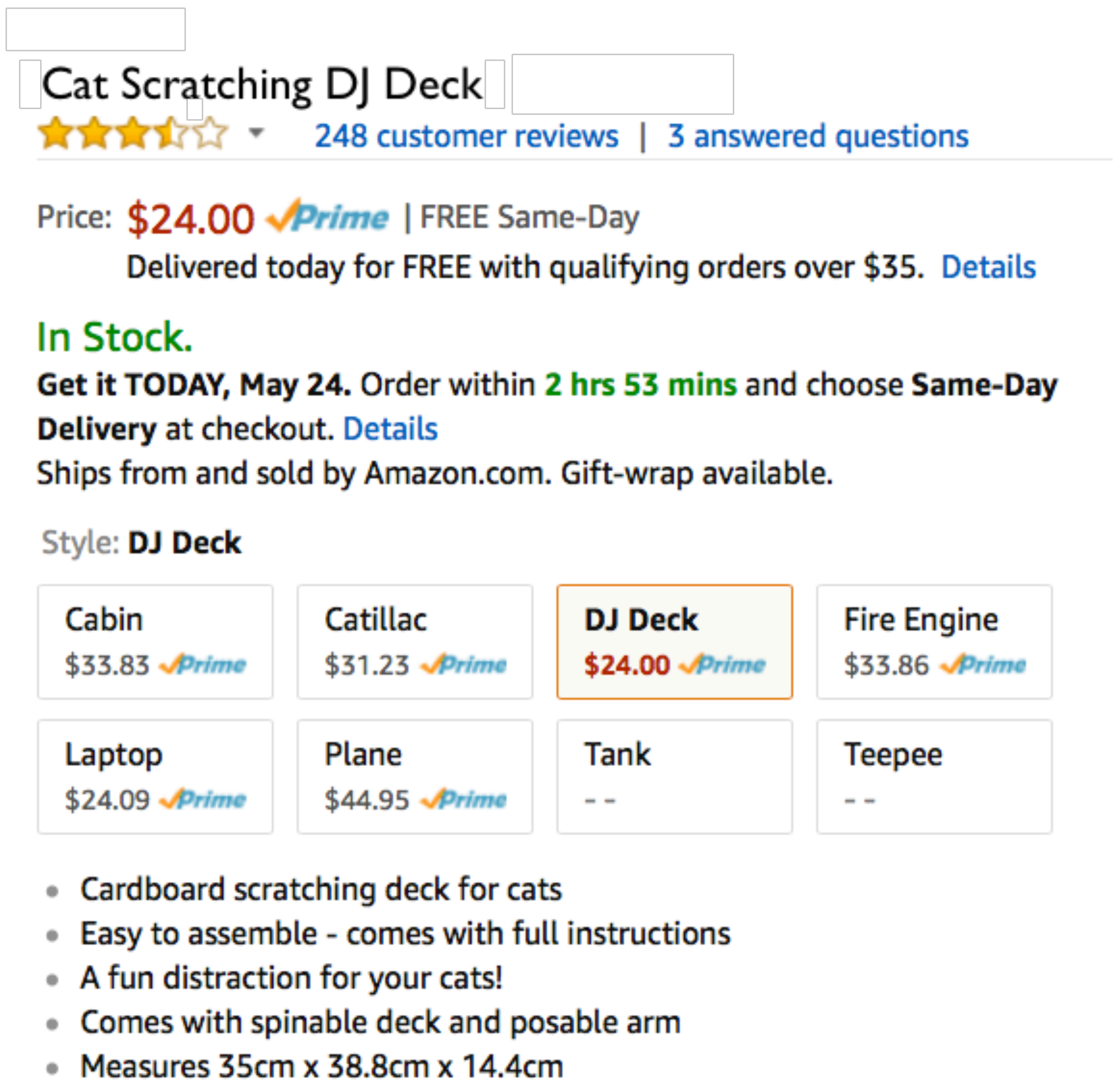
Favorite Medium Cat Tunnel Foldable Lightweight Fun Dangling Ball Toy,...
★★★★½ 124
$9.99 Prime

PET SHINEWINGS- Basic Cat Scratcher Scratching Board Pad
★★★★★ 1
$23.99 Prime

- What product might this be related to?

Other Real Life Uses for Graphs

Roll over image to zoom in



Cat Scratching DJ Deck

248 customer reviews | 3 answered questions

Price: $24.00 Prime | FREE Same-Day
Delivered today for FREE with qualifying orders over $35. Details

In Stock.

Get it TODAY, May 24. Order within 2 hrs 53 mins and choose **Same-Day Delivery** at checkout. Details
Ships from and sold by Amazon.com. Gift-wrap available.

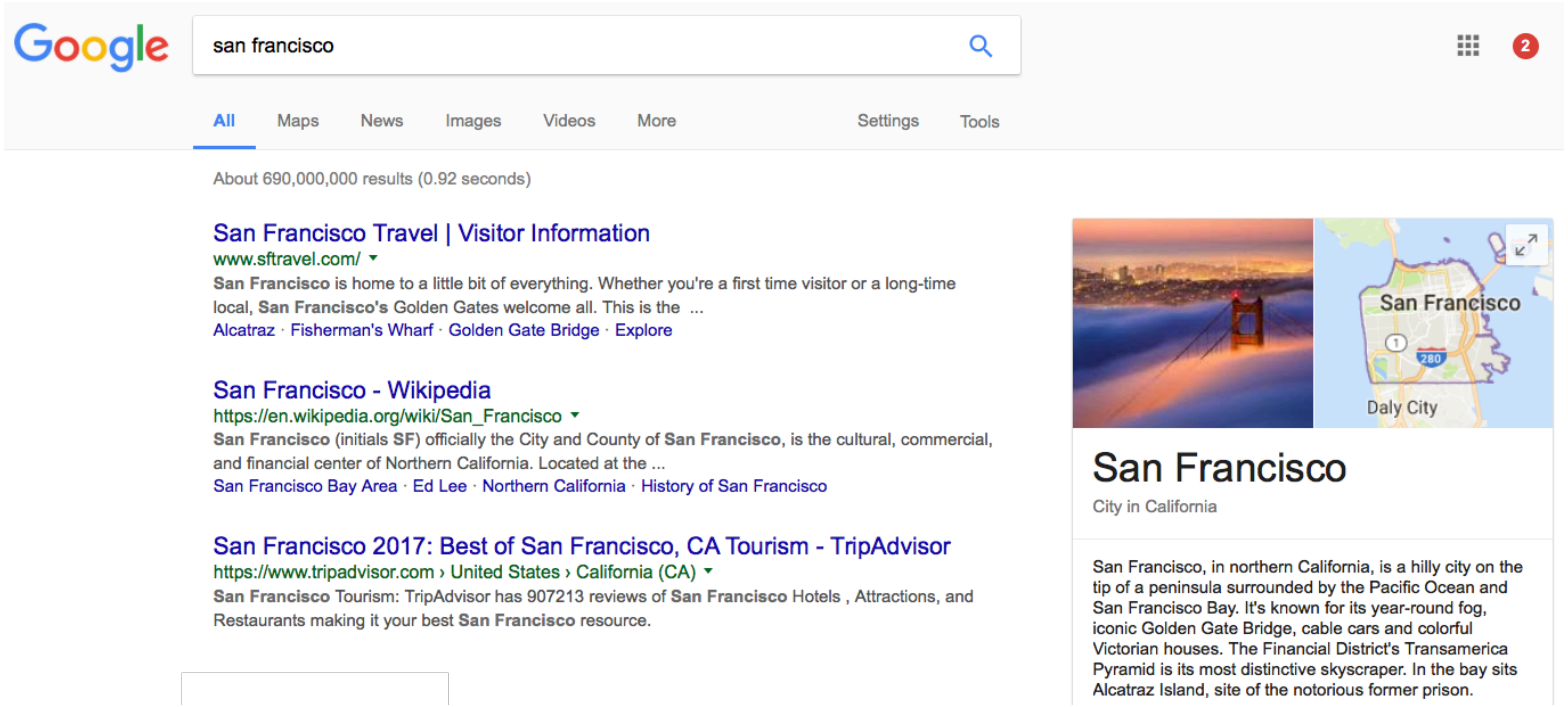
Style: **DJ Deck**

Cabin $33.83 Prime	Catillac $31.23 Prime	**DJ Deck** $24.00 Prime	Fire Engine $33.86 Prime
Laptop $24.09 Prime	Plane $44.95 Prime	Tank - -	Teepee - -

- Cardboard scratching deck for cats
- Easy to assemble - comes with full instructions
- A fun distraction for your cats!
- Comes with spinable deck and posable arm
- Measures 35cm x 38.8cm x 14.4cm

Other Real Life Uses for Graphs

- Web page searching (discussed last week -- Wikipedia path to Philosophy)
- Google Maps (Trailblazer!)



Google

san francisco

All Maps News Images Videos More Settings Tools

About 690,000,000 results (0.92 seconds)

San Francisco Travel | Visitor Information
www.sftravel.com/
San Francisco is home to a little bit of everything. Whether you're a first time visitor or a long-time local, **San Francisco's** Golden Gates welcome all. This is the ...
Alcatraz · Fisherman's Wharf · Golden Gate Bridge · Explore

San Francisco - Wikipedia
https://en.wikipedia.org/wiki/San_Francisco
San Francisco (initials **SF**) officially the City and County of **San Francisco**, is the cultural, commercial, and financial center of Northern California. Located at the ...
San Francisco Bay Area · Ed Lee · Northern California · History of San Francisco

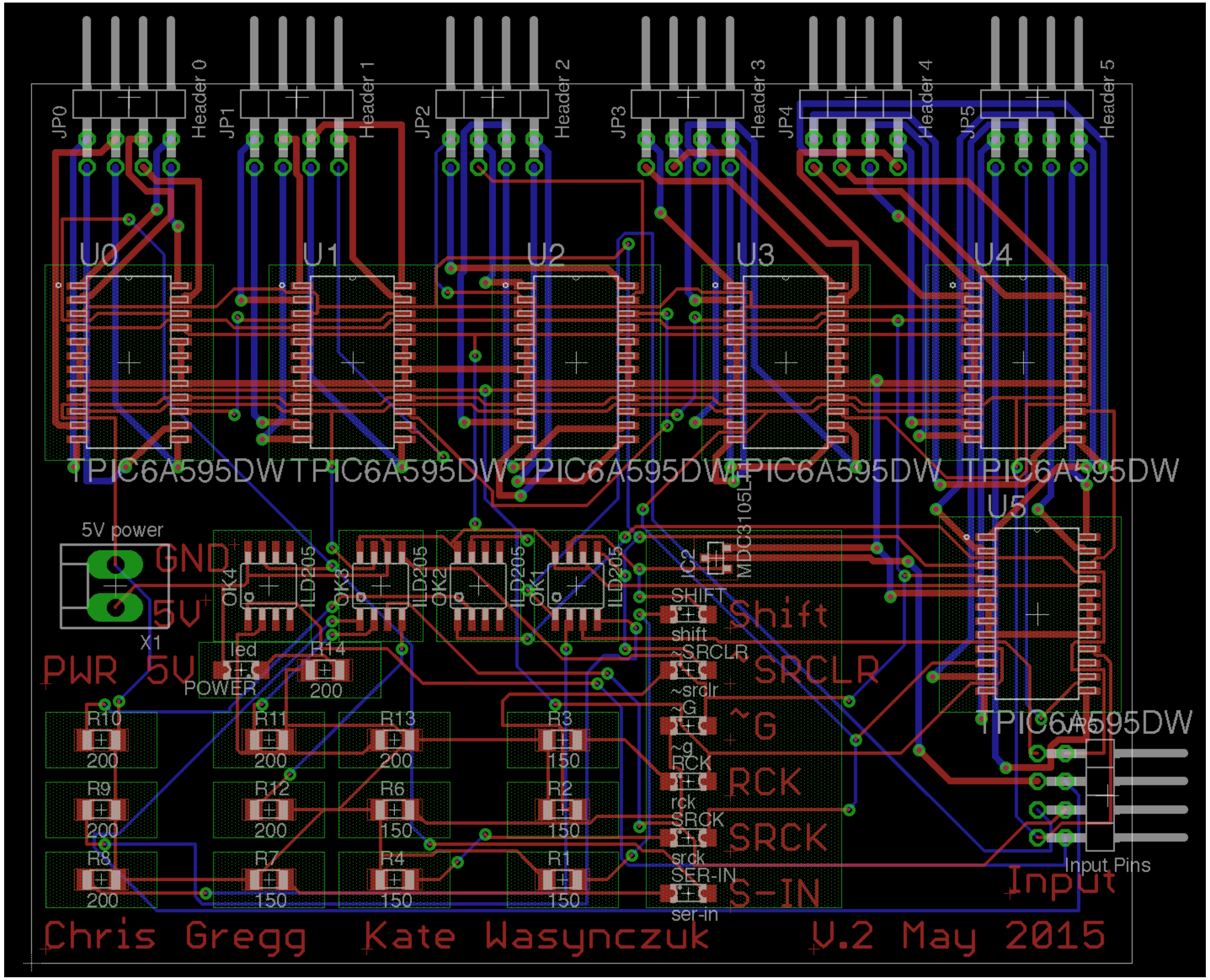
San Francisco 2017: Best of San Francisco, CA Tourism - TripAdvisor
https://www.tripadvisor.com › United States › California (CA)
San Francisco Tourism: TripAdvisor has 907213 reviews of **San Francisco** Hotels , Attractions, and Restaurants making it your best **San Francisco** resource.

San Francisco
City in California

San Francisco, in northern California, is a hilly city on the tip of a peninsula surrounded by the Pacific Ocean and San Francisco Bay. It's known for its year-round fog, iconic Golden Gate Bridge, cable cars and colorful Victorian houses. The Financial District's Transamerica Pyramid is its most distinctive skyscraper. In the bay sits Alcatraz Island, site of the notorious former prison.

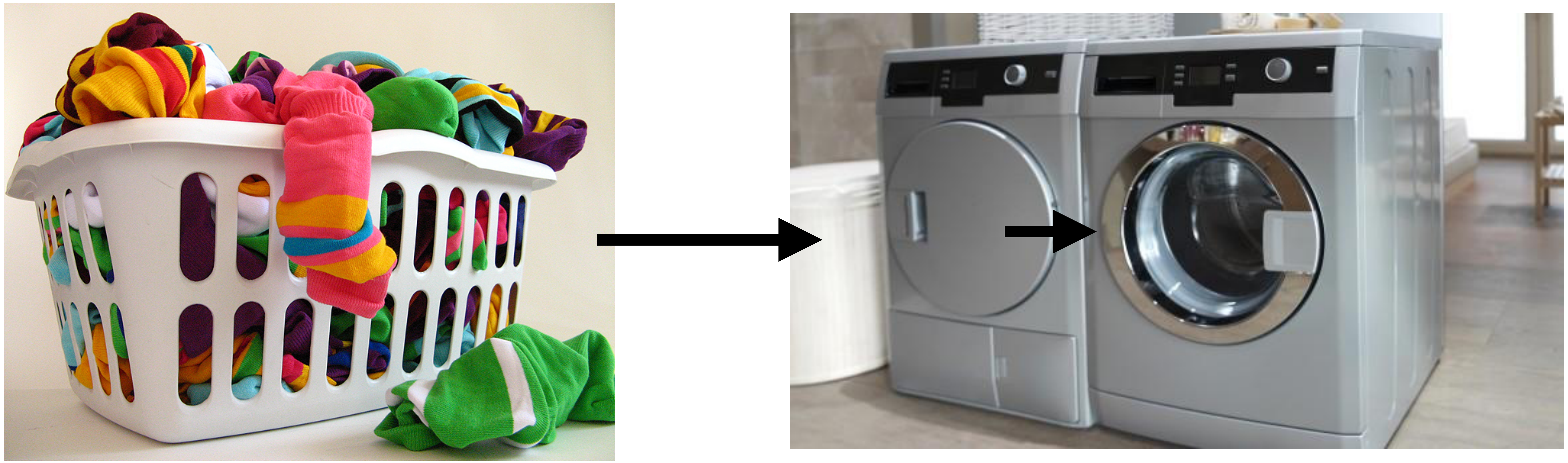
Other Real Life Uses for Graphs

- Routing circuits:



Other Real Life Uses for Graphs

- Scheduling work based on dependencies (e.g., when doing laundry, the washer must finish before the dryer, and before folding) -- this is called a "topological sort")



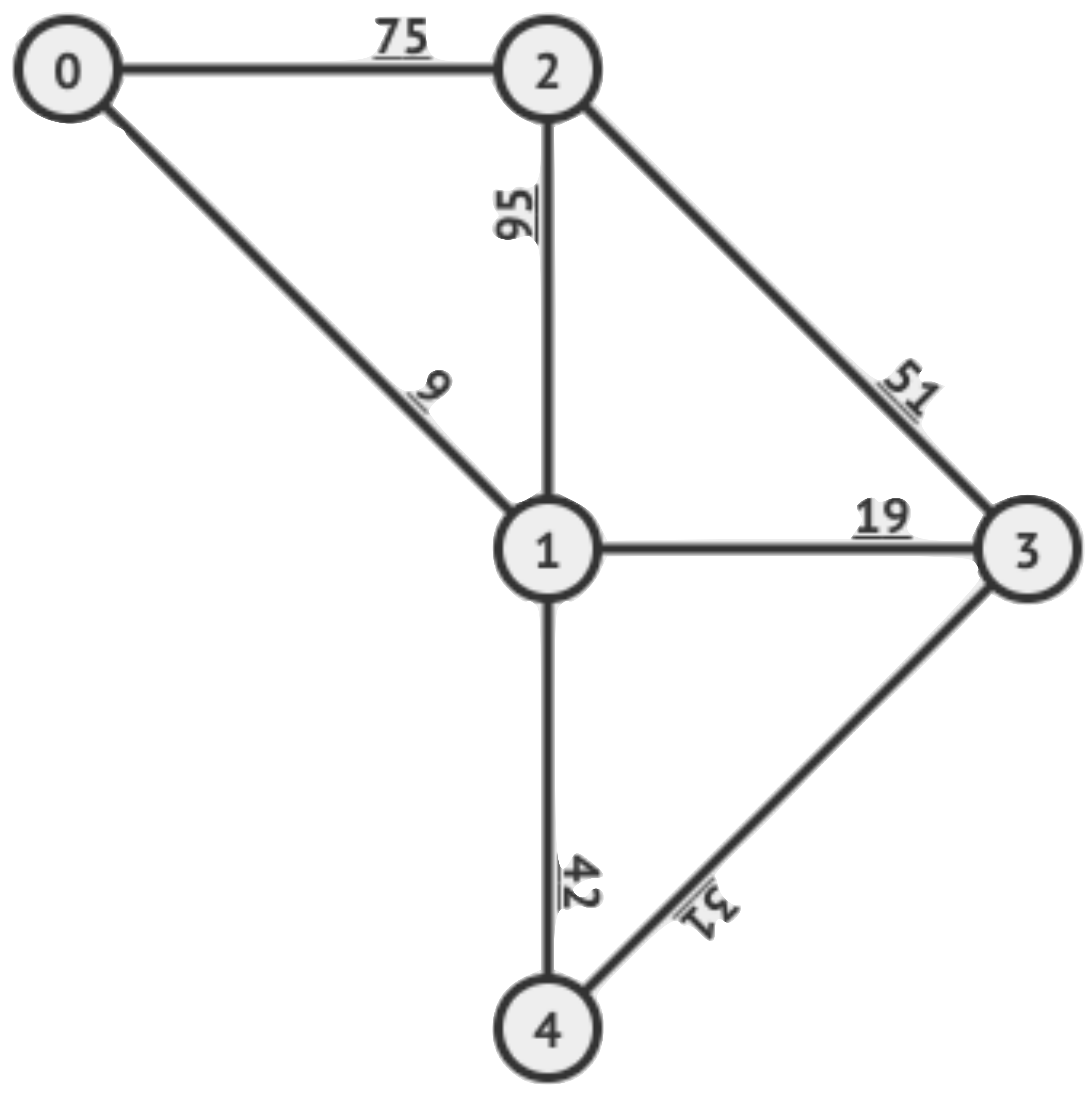
$$\begin{tikzcd}
b \arrow[r] \arrow[rrrrrr, bend left] & g \arrow[r] \arrow[rrrr, bend right] & k \arrow[r] & d \arrow[rr, bend left] & a \arrow[r] \arrow[rrrr, bend left] & e \arrow[r] \arrow[rr, bend right] & h & f \arrow[r] & c
\end{tikzcd}$$

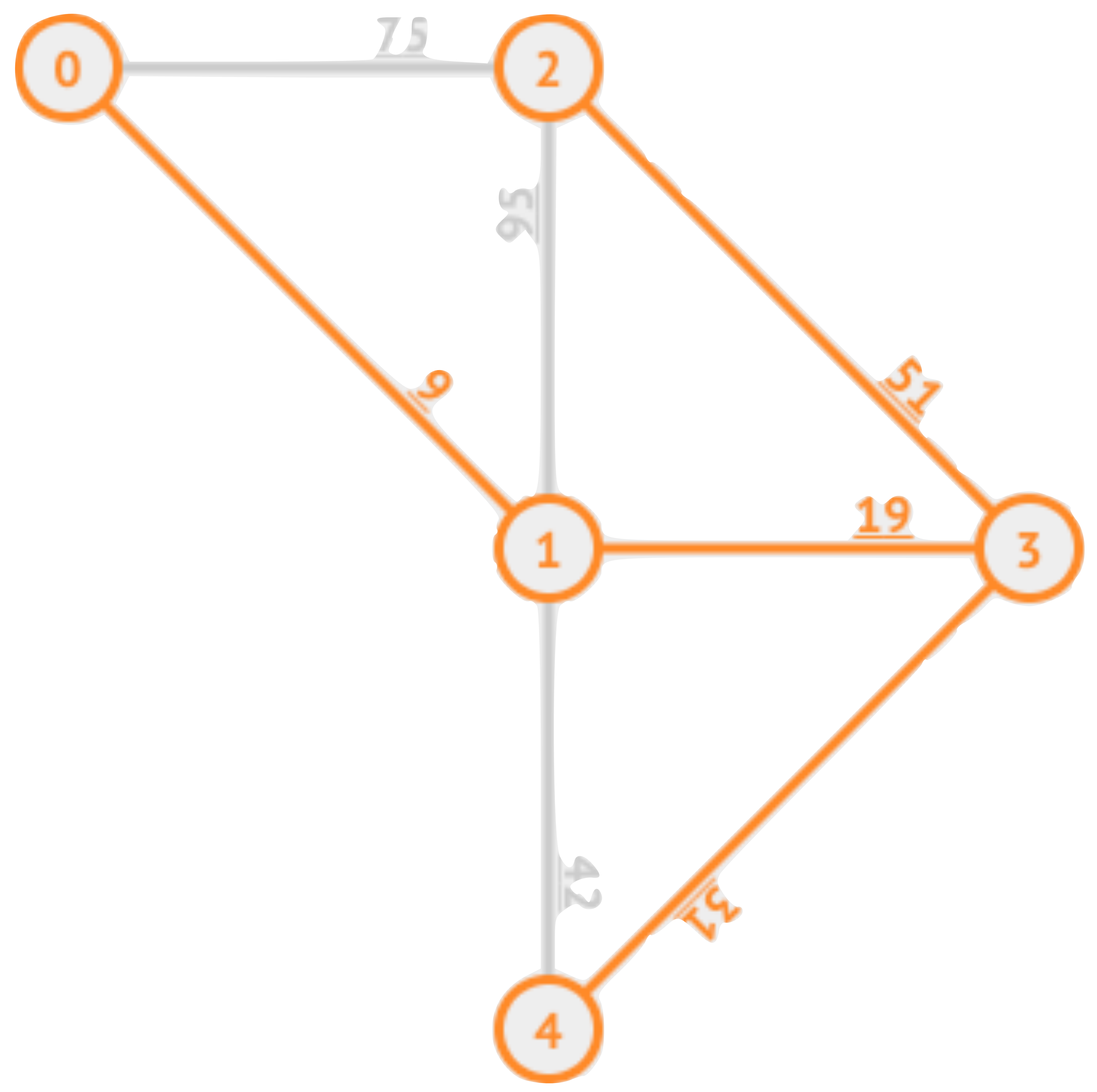
Other Real Life Uses for Graphs

- The "Oracle of Bacon": https://oracleofbacon.org (just graph searching!)

Other Real Life Uses for Graphs

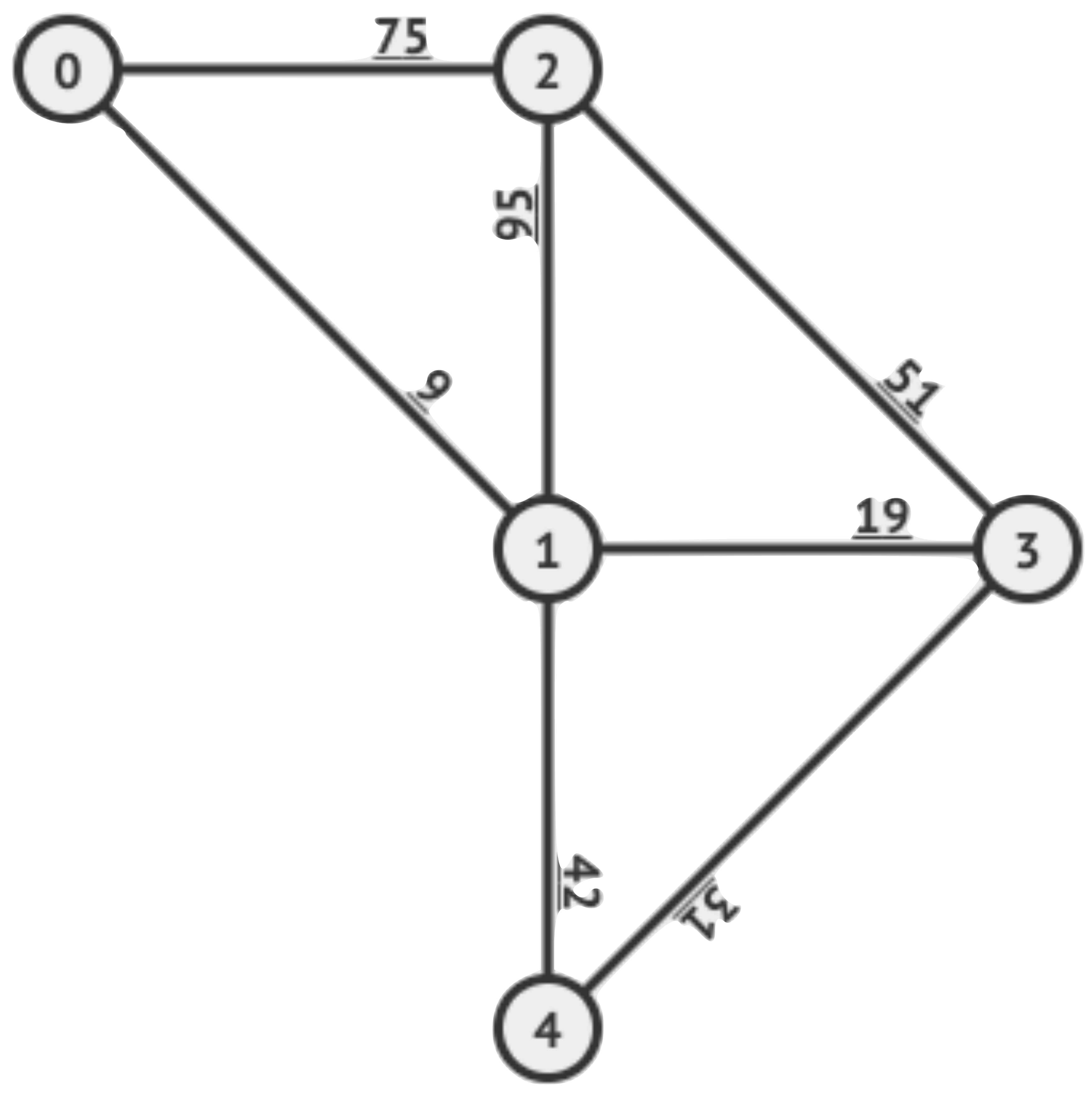
- Telecommunications: find the least expensive way to lay out a set of cables for a telephone or cable TV system ("a Minimum Spanning Tree")

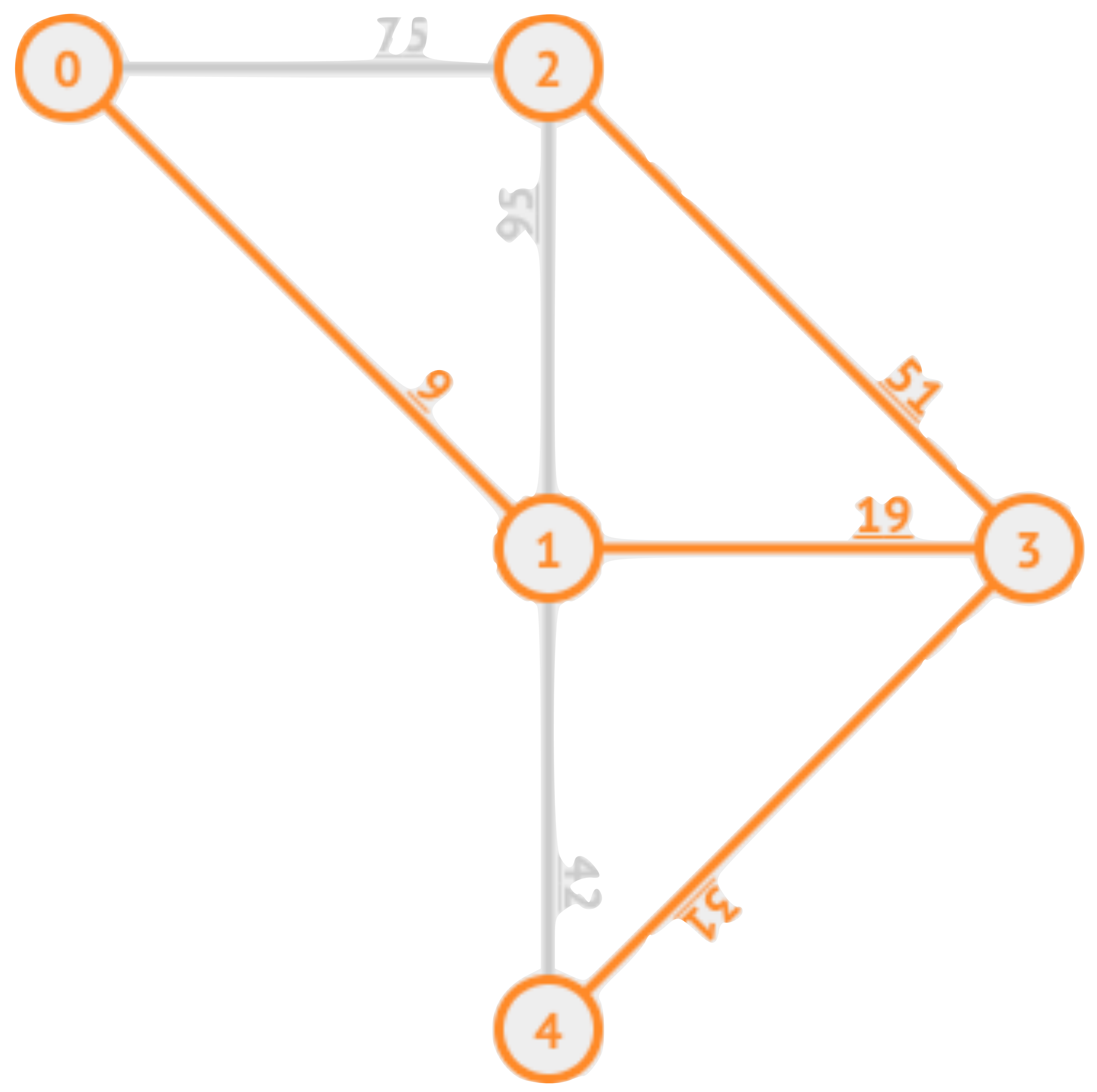




Spanning Trees and Minimum Spanning Trees 🔑

Definition: A **Spanning Tree (ST)** of a connected undirected weighted graph **G** is a subgraph of **G** that is a **tree** and **connects (spans) all vertices of G**. A graph **G** can have multiple STs. A **Minimum Spanning Tree (MST)** of **G** is a ST of **G** that has the **smallest total weight** among the various STs. A graph **G** can have multiple MSTs but the MST weight is unique.





Minimum Spanning Tree

Kruskal's Algorithm to find a Minimum Spanning Tree

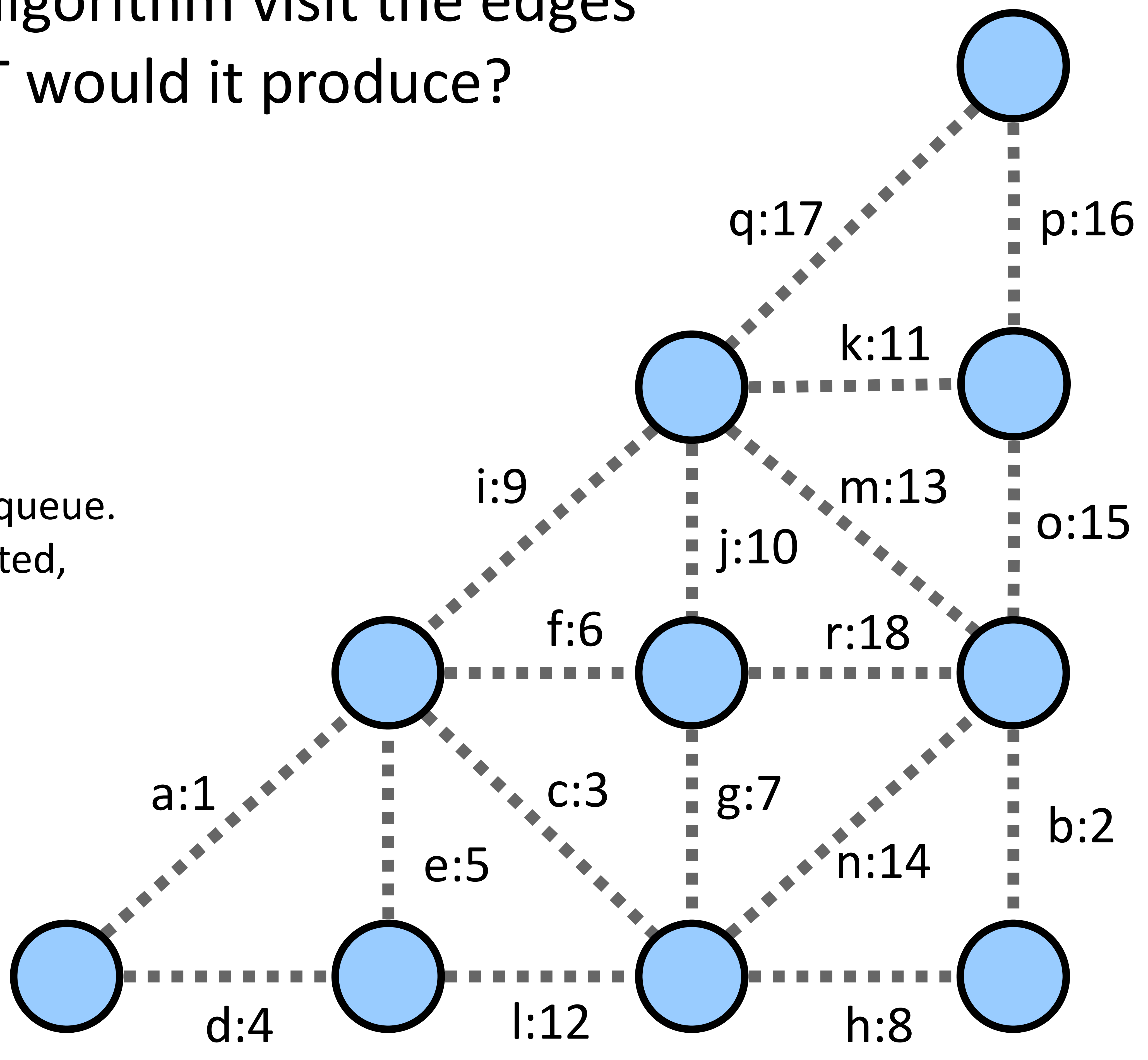
- **Kruskal's algorithm**: Finds a MST in a given graph.

```
function kruskal(graph):
  Remove all edges from the graph.
  Place all edges into a priority queue based on their weight (cost).
  While the priority queue is not empty:
      Dequeue an edge e from the priority queue.
      If e's endpoints aren't already connected to one another,
          add that edge into the graph.
      Otherwise, skip the edge.
```

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
             Dequeue an edge e from the priority queue.
             If e's endpoints aren't already connected,
                add that edge into the graph.
             Otherwise, skip the edge.
```

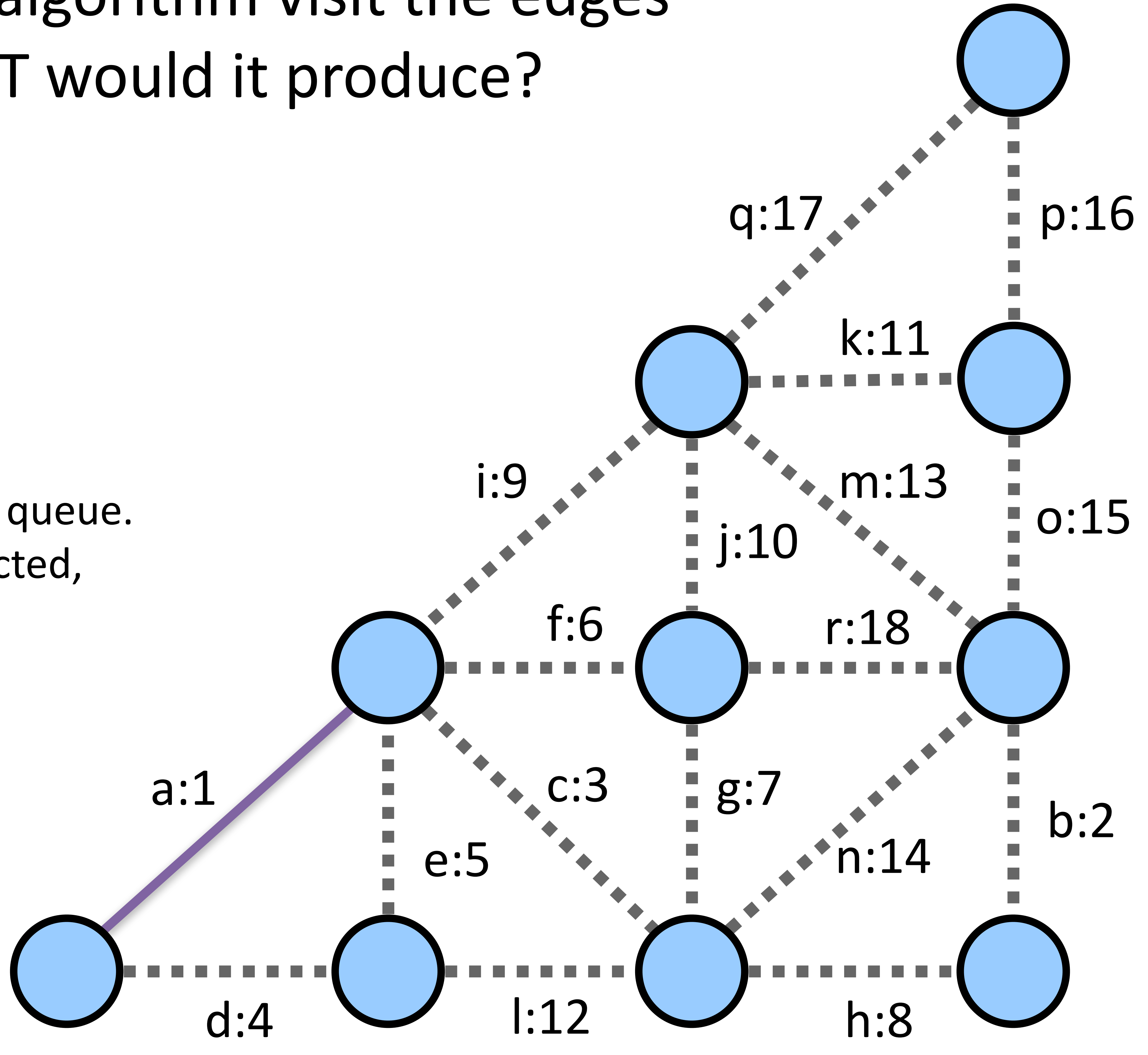


pq = {a:1, b:2, c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
             Dequeue an edge e from the priority queue.
             If e's endpoints aren't already connected,
                add that edge into the graph.
             Otherwise, skip the edge.
```

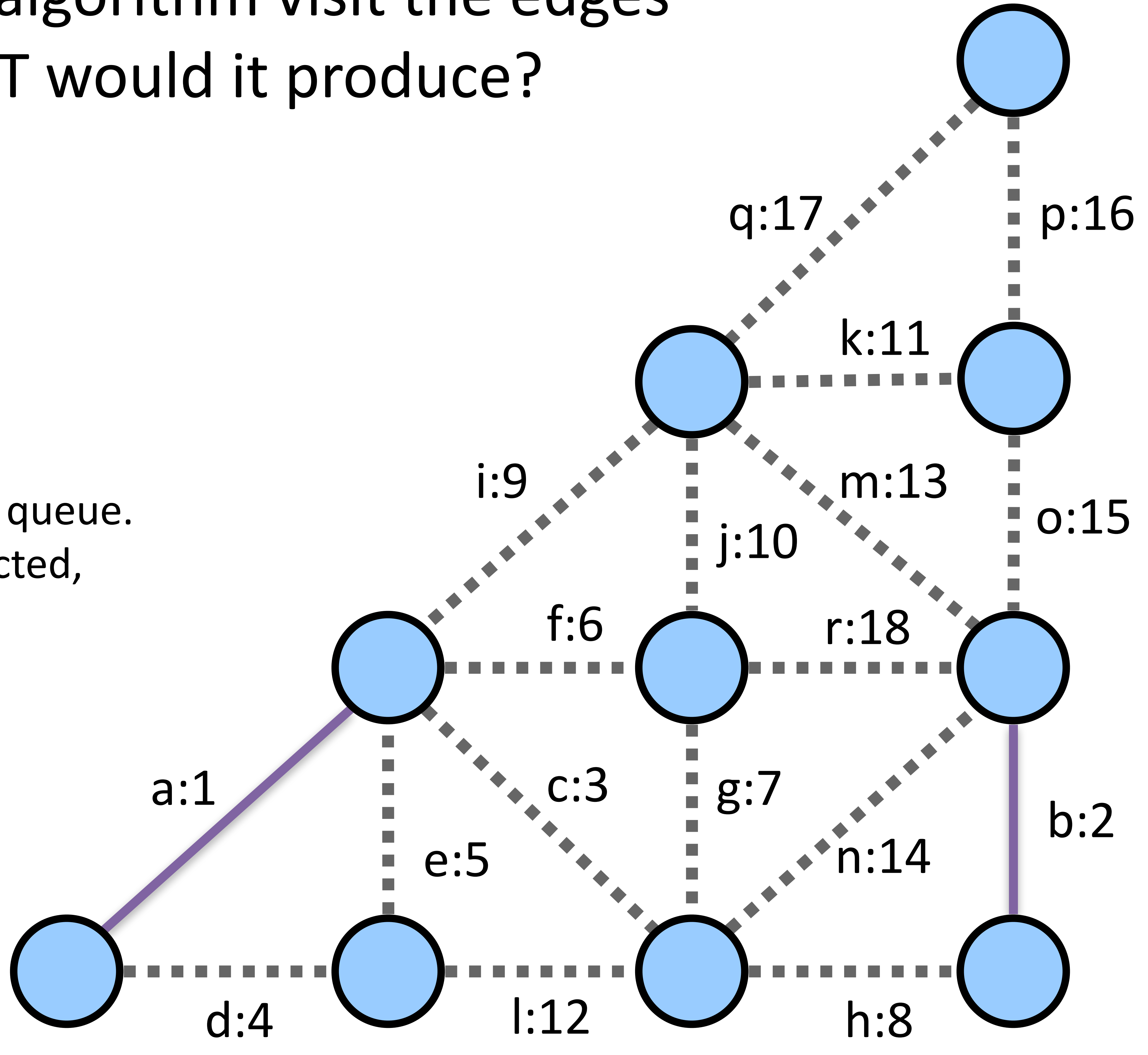


pq = {**a:1**, b:2, c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
           Dequeue an edge e from the priority queue.
           If e's endpoints aren't already connected,
              add that edge into the graph.
           Otherwise, skip the edge.
```

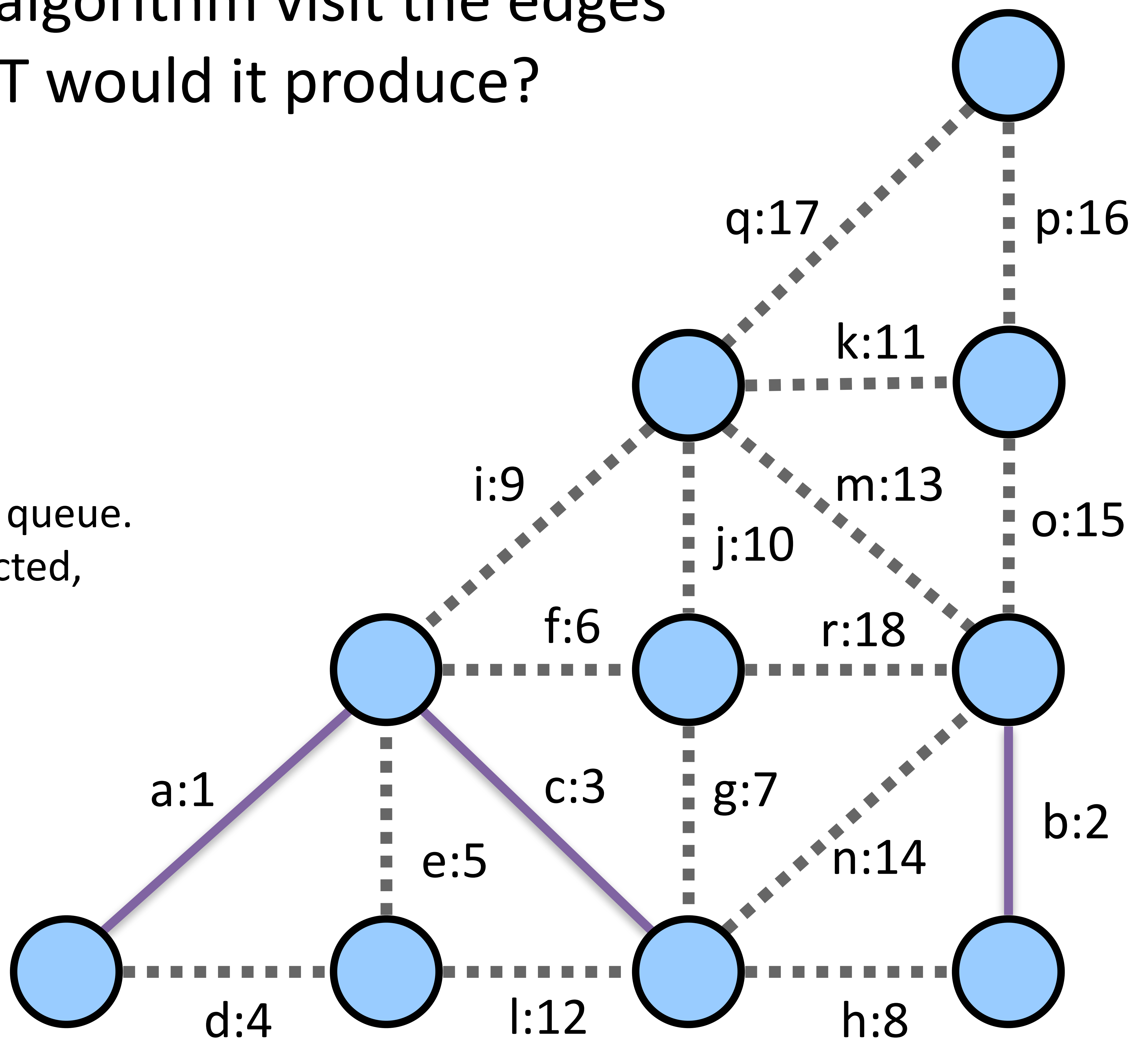


pq = {**b:2**, c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
  Remove all edges from the graph.
  Place all edges into a priority queue
      based on their weight (cost).
  While the priority queue is not empty:
      Dequeue an edge e from the priority queue.
      If e's endpoints aren't already connected,
        add that edge into the graph.
      Otherwise, skip the edge.
```

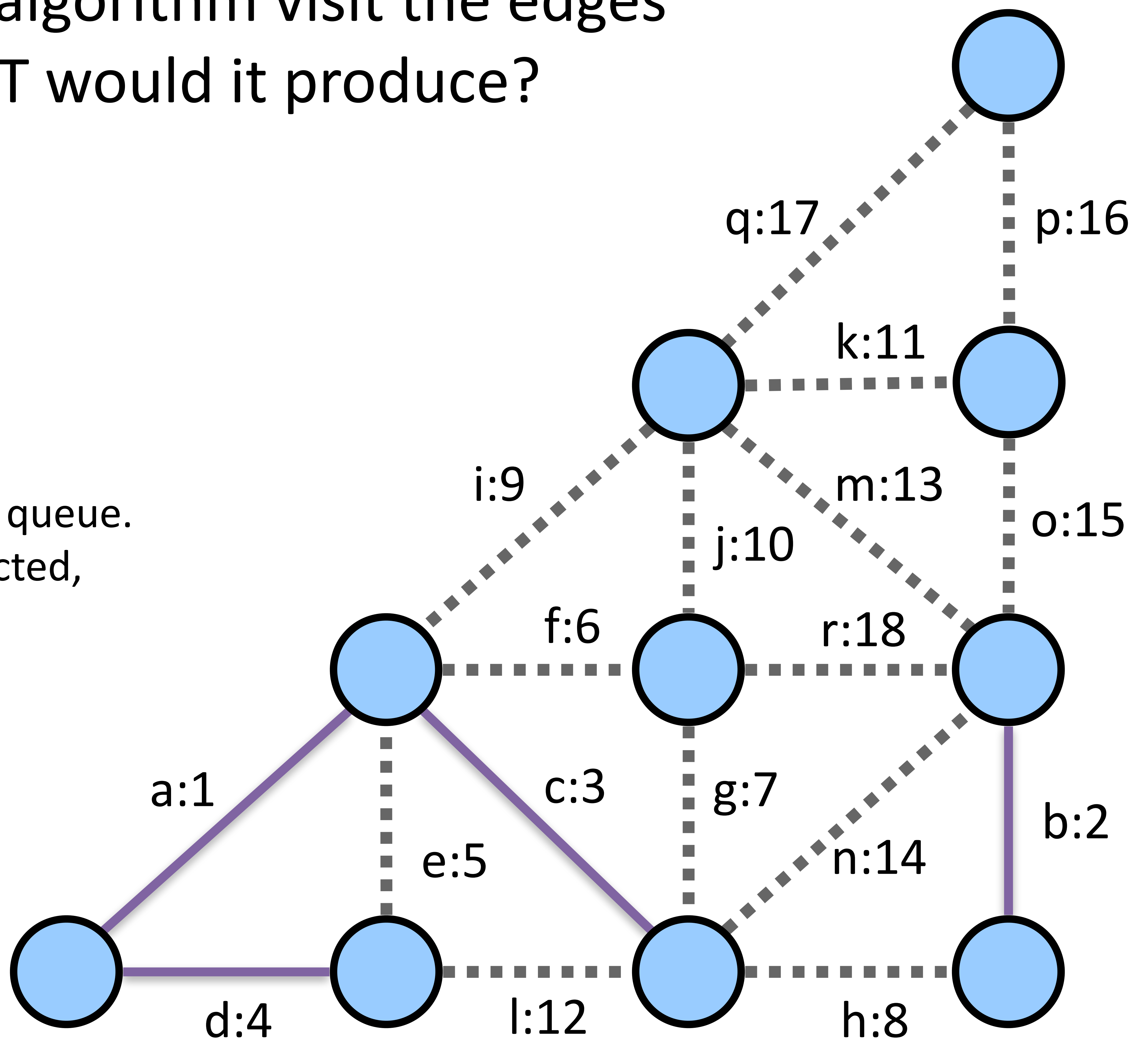


pq = {**c:3**, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
       Dequeue an edge e from the priority queue.
       If e's endpoints aren't already connected,
          add that edge into the graph.
       Otherwise, skip the edge.
```

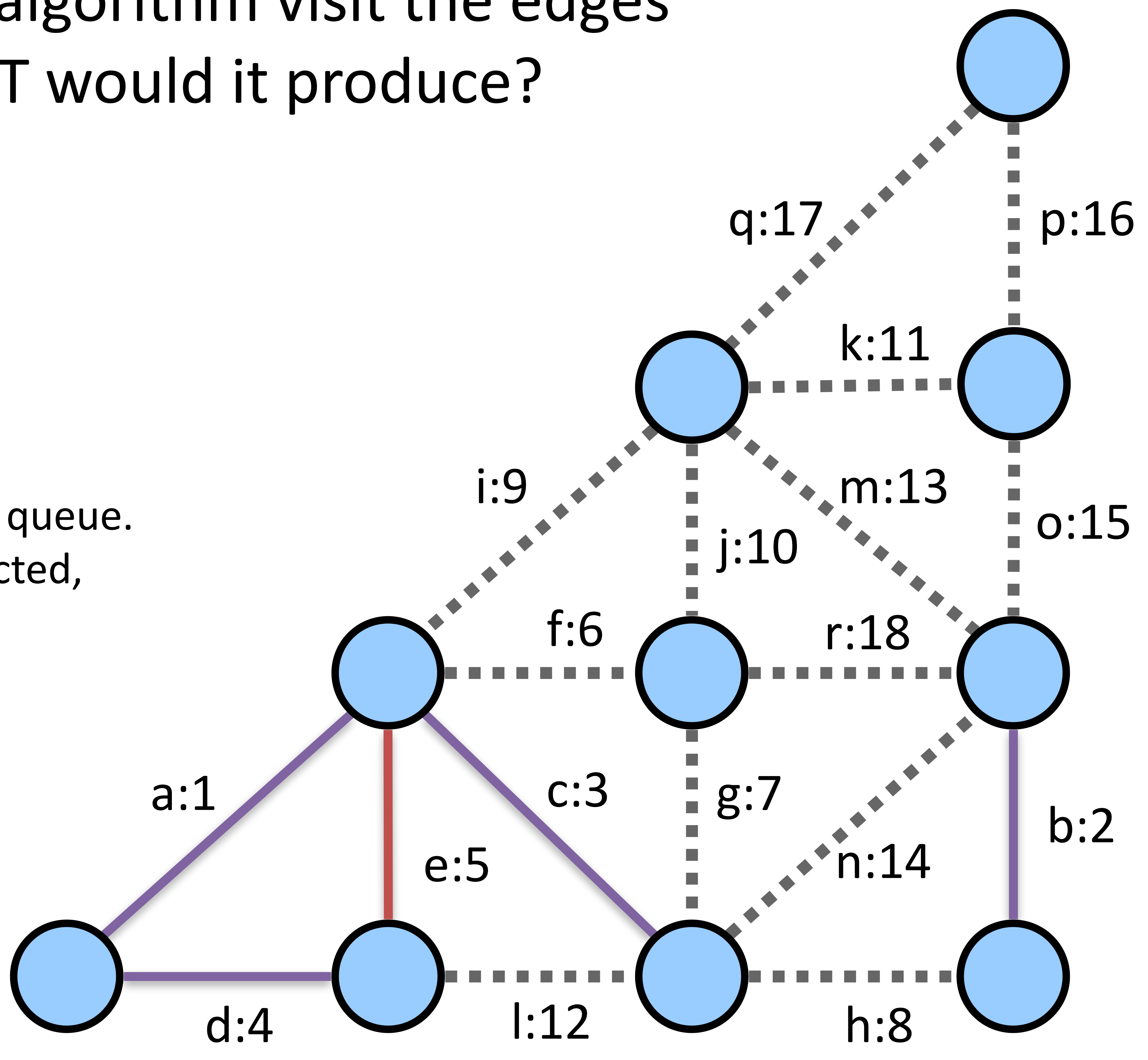


pq = {**d:4**, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
             Dequeue an edge e from the priority queue.
             If e's endpoints aren't already connected,
                add that edge into the graph.
             Otherwise, skip the edge.
```

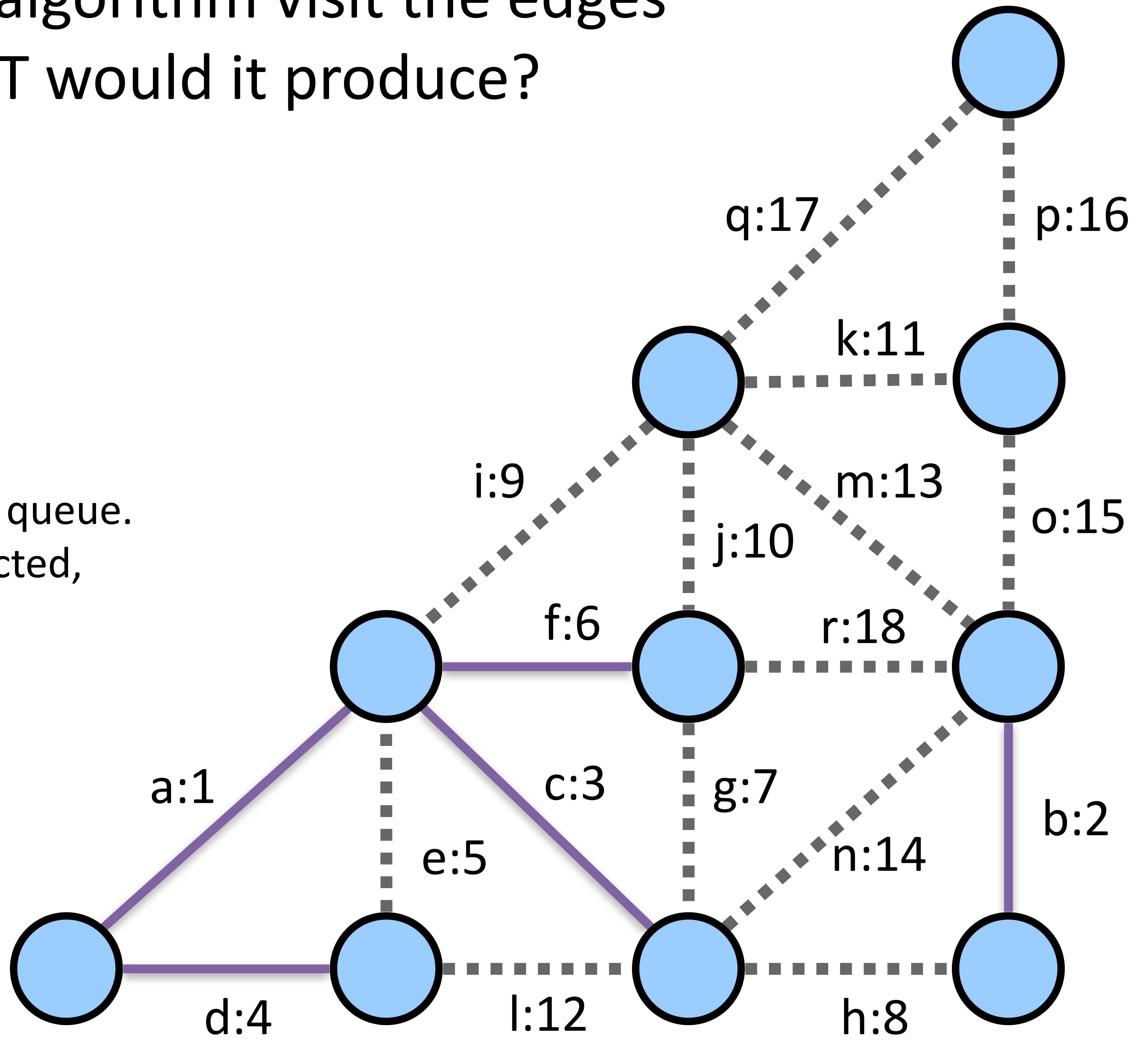


pq = {**e:5**, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
             Dequeue an edge e from the priority queue.
             If e's endpoints aren't already connected,
                add that edge into the graph.
             Otherwise, skip the edge.
```

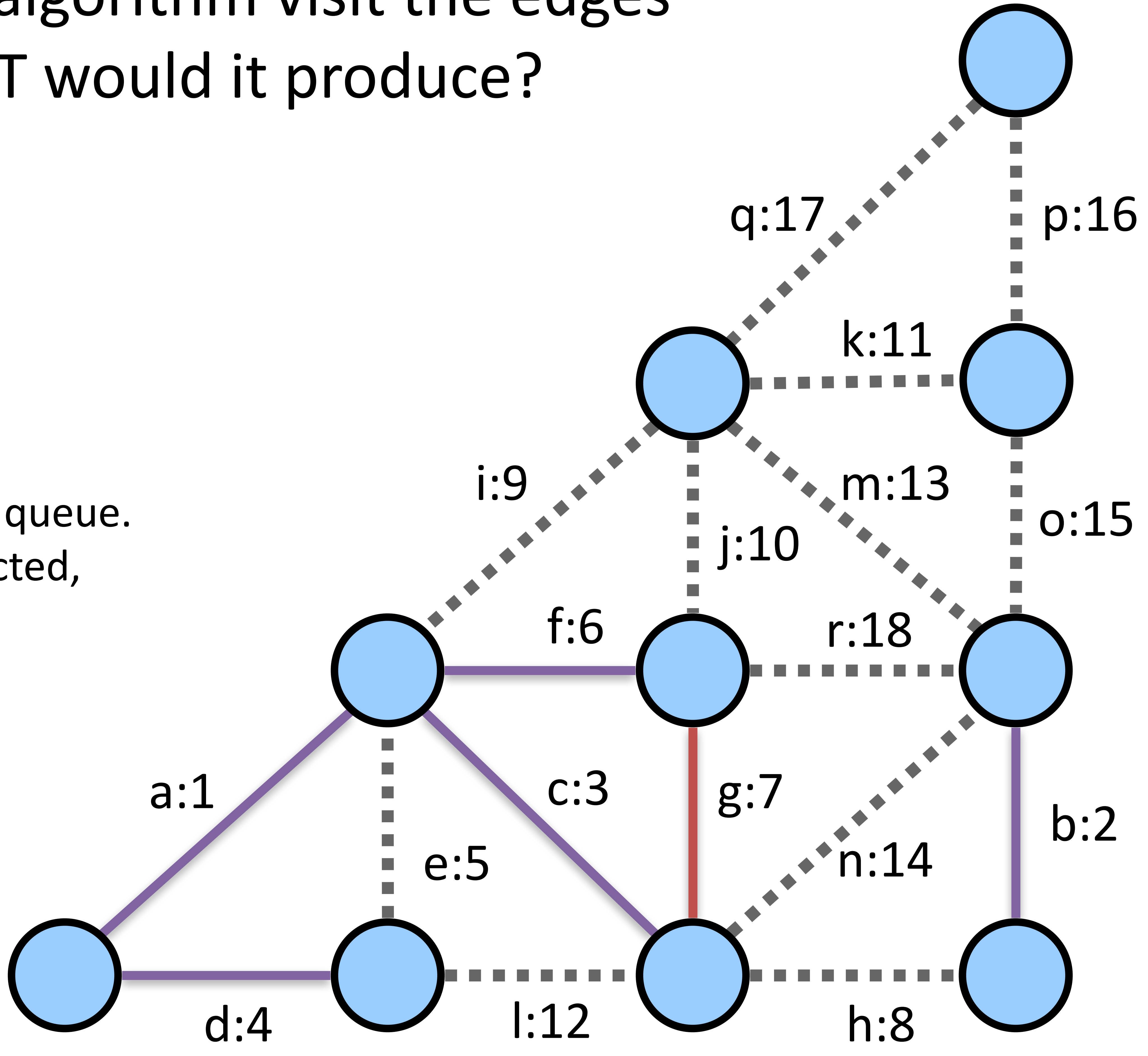


pq = {**f:6**, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
       Dequeue an edge e from the priority queue.
       If e's endpoints aren't already connected,
          add that edge into the graph.
       Otherwise, skip the edge.
```

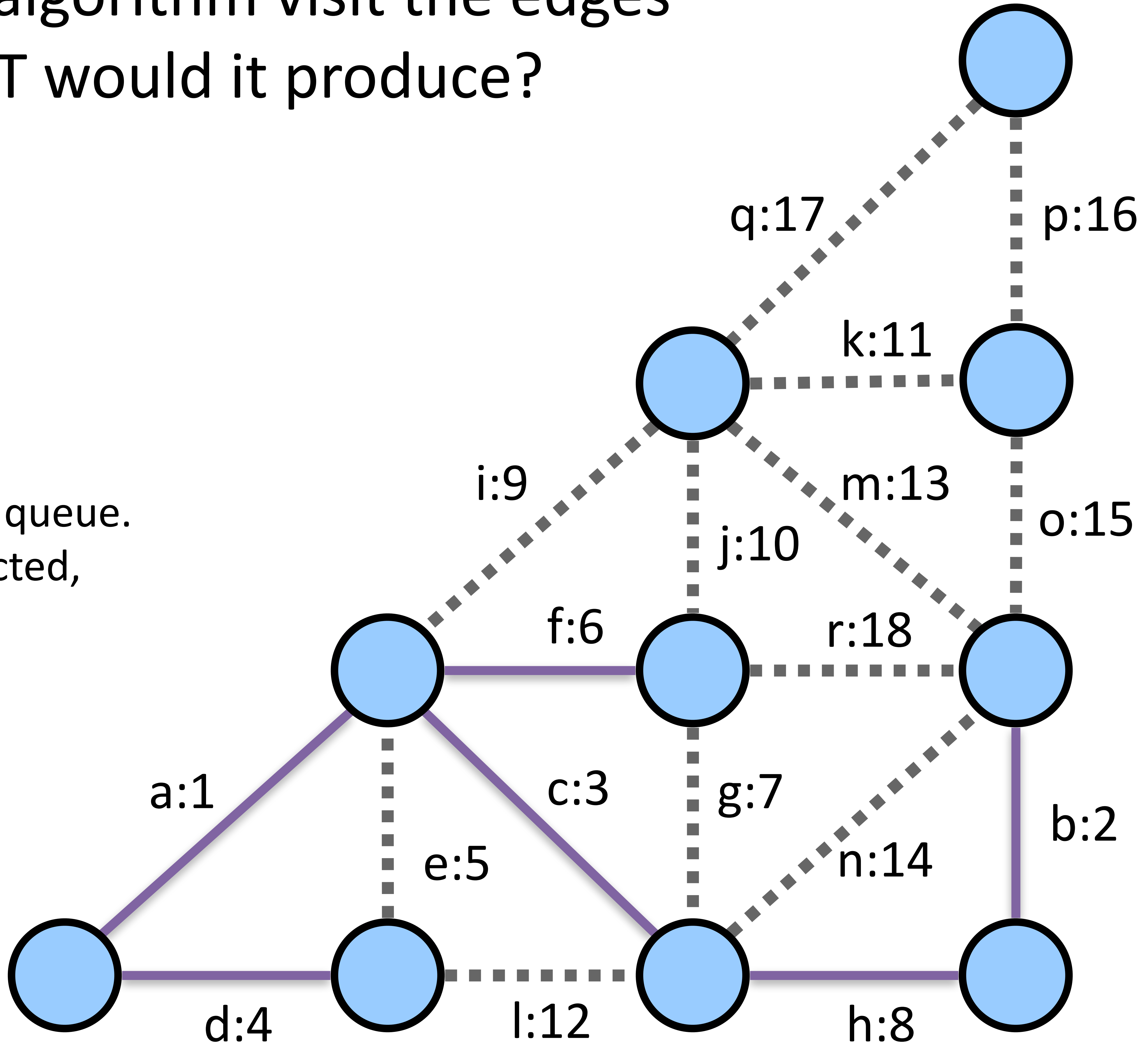


pq = {**g:7**, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
             Dequeue an edge e from the priority queue.
             If e's endpoints aren't already connected,
                add that edge into the graph.
             Otherwise, skip the edge.
```

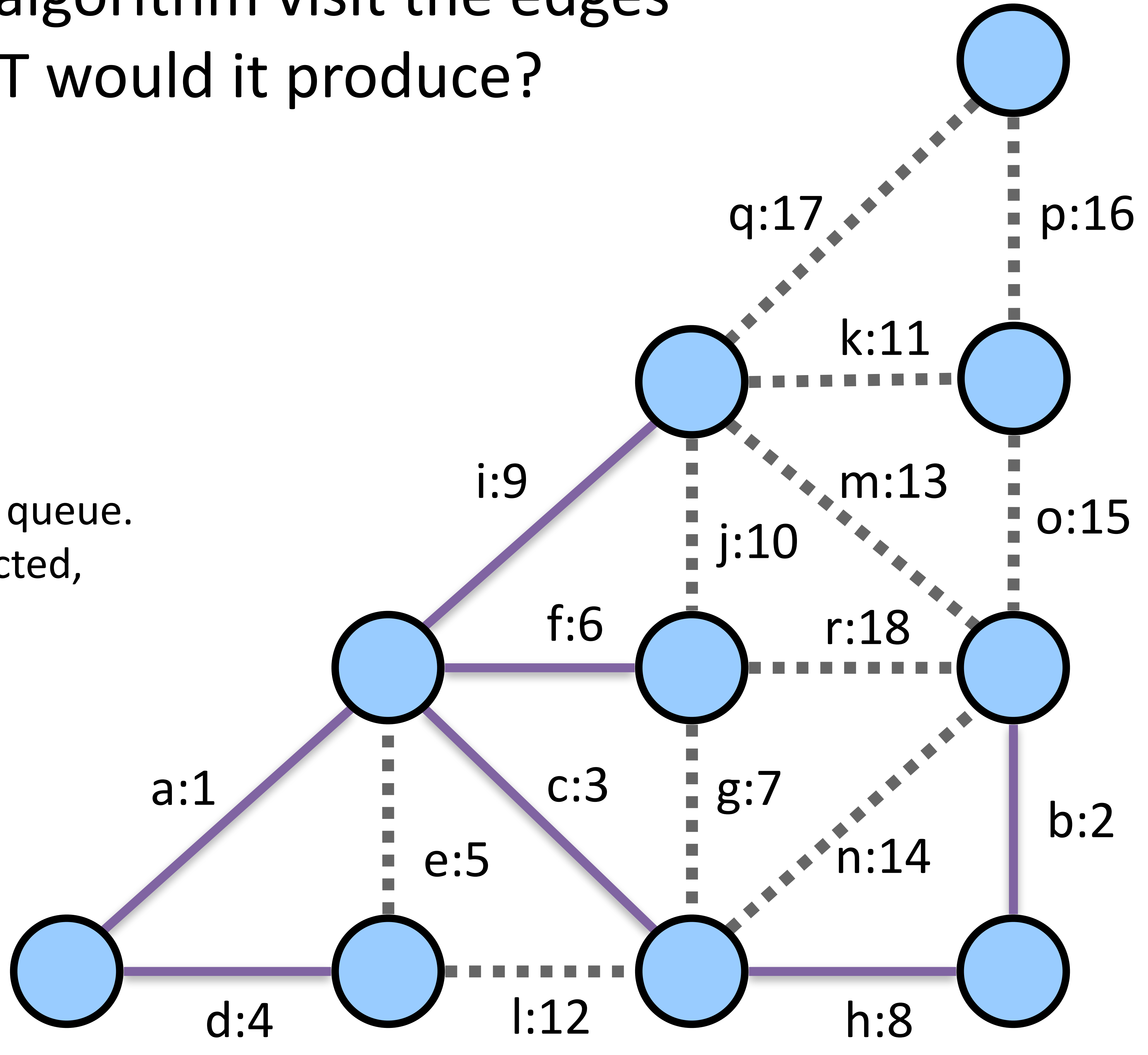


pq = {**h:8**, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
       Dequeue an edge e from the priority queue.
       If e's endpoints aren't already connected,
          add that edge into the graph.
       Otherwise, skip the edge.
```

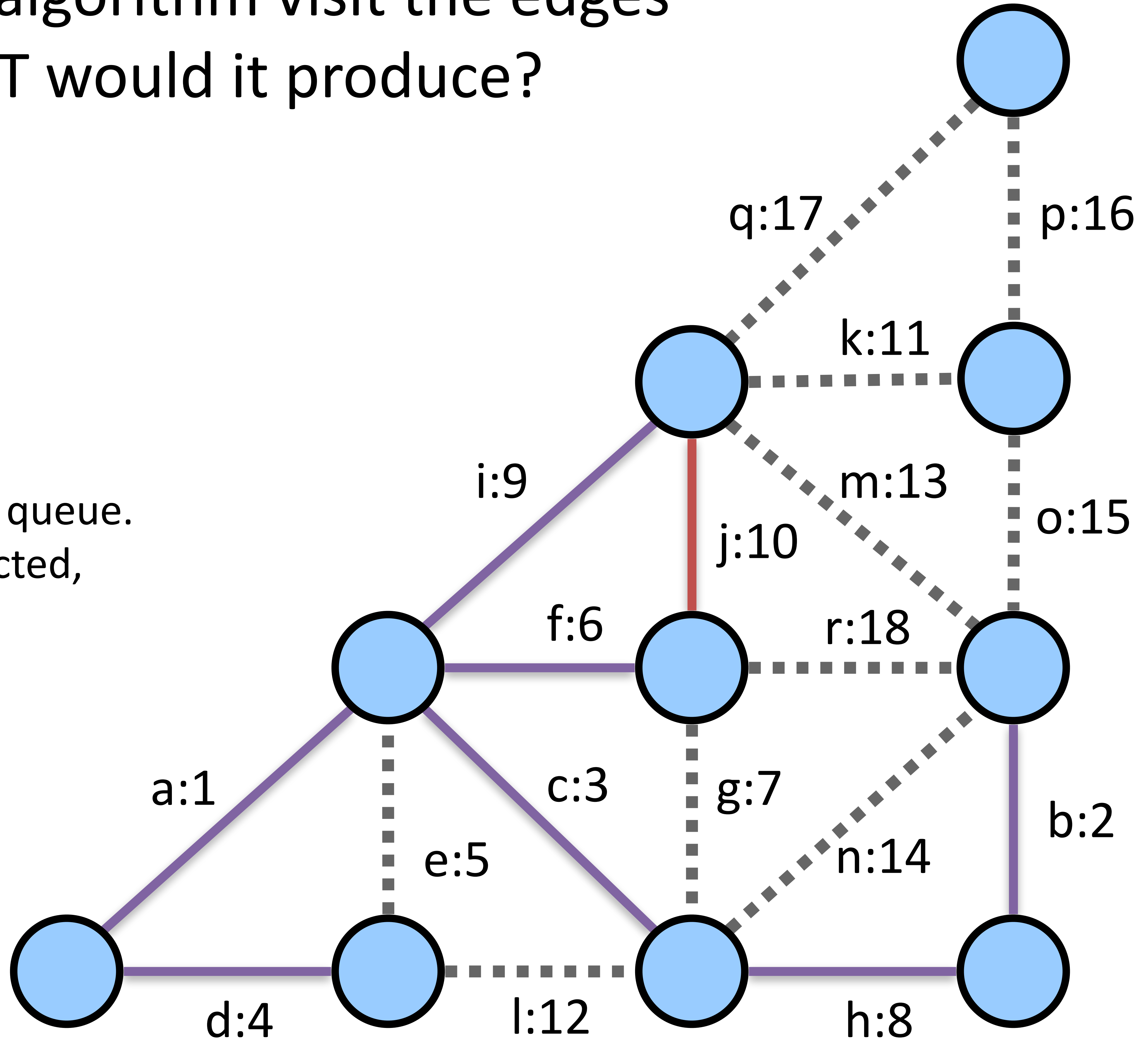


pq = {**i:9**, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
           Dequeue an edge e from the priority queue.
           If e's endpoints aren't already connected,
              add that edge into the graph.
           Otherwise, skip the edge.
```

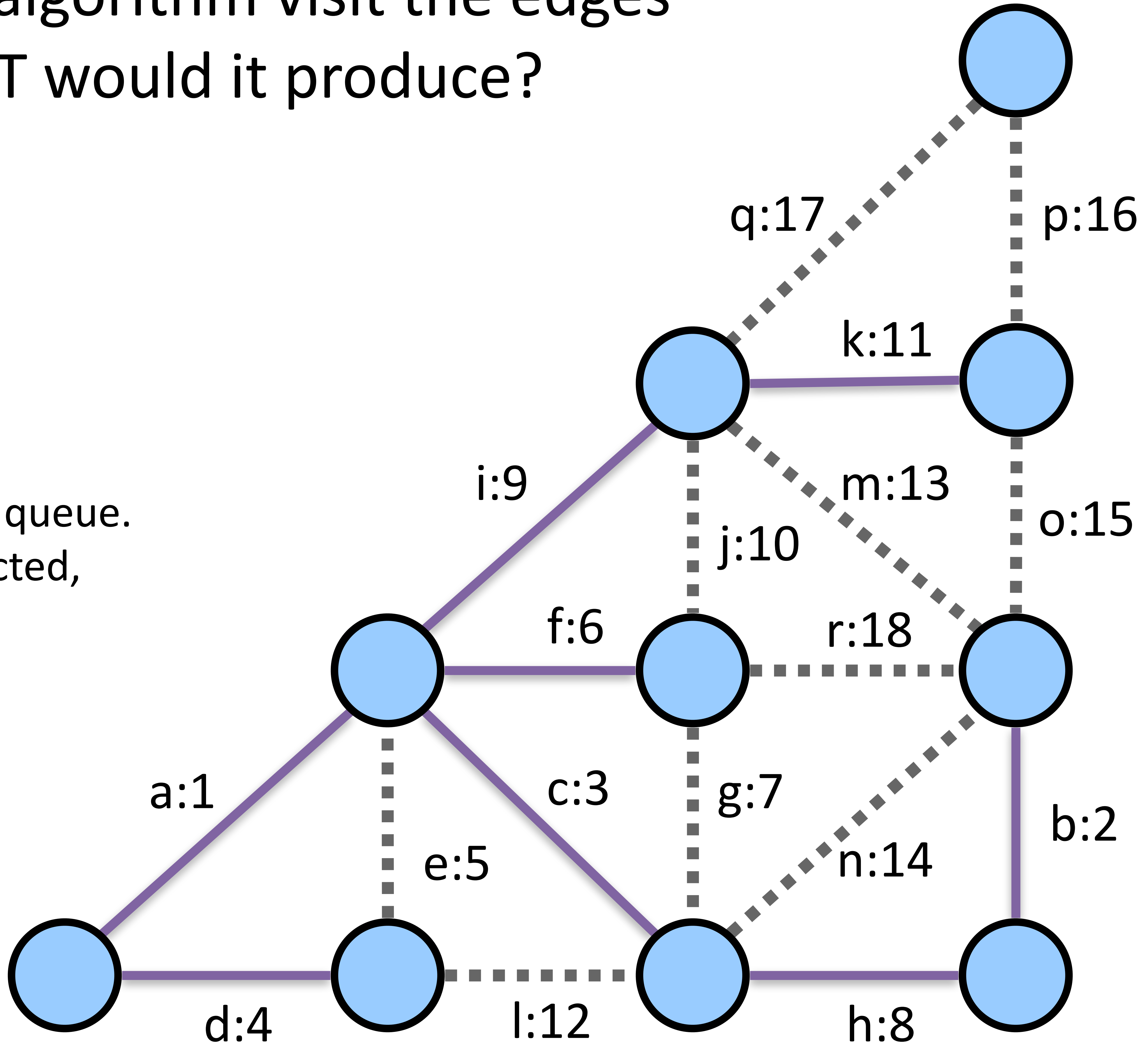


pq = {**j:10**, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
             Dequeue an edge e from the priority queue.
             If e's endpoints aren't already connected,
                add that edge into the graph.
             Otherwise, skip the edge.
```

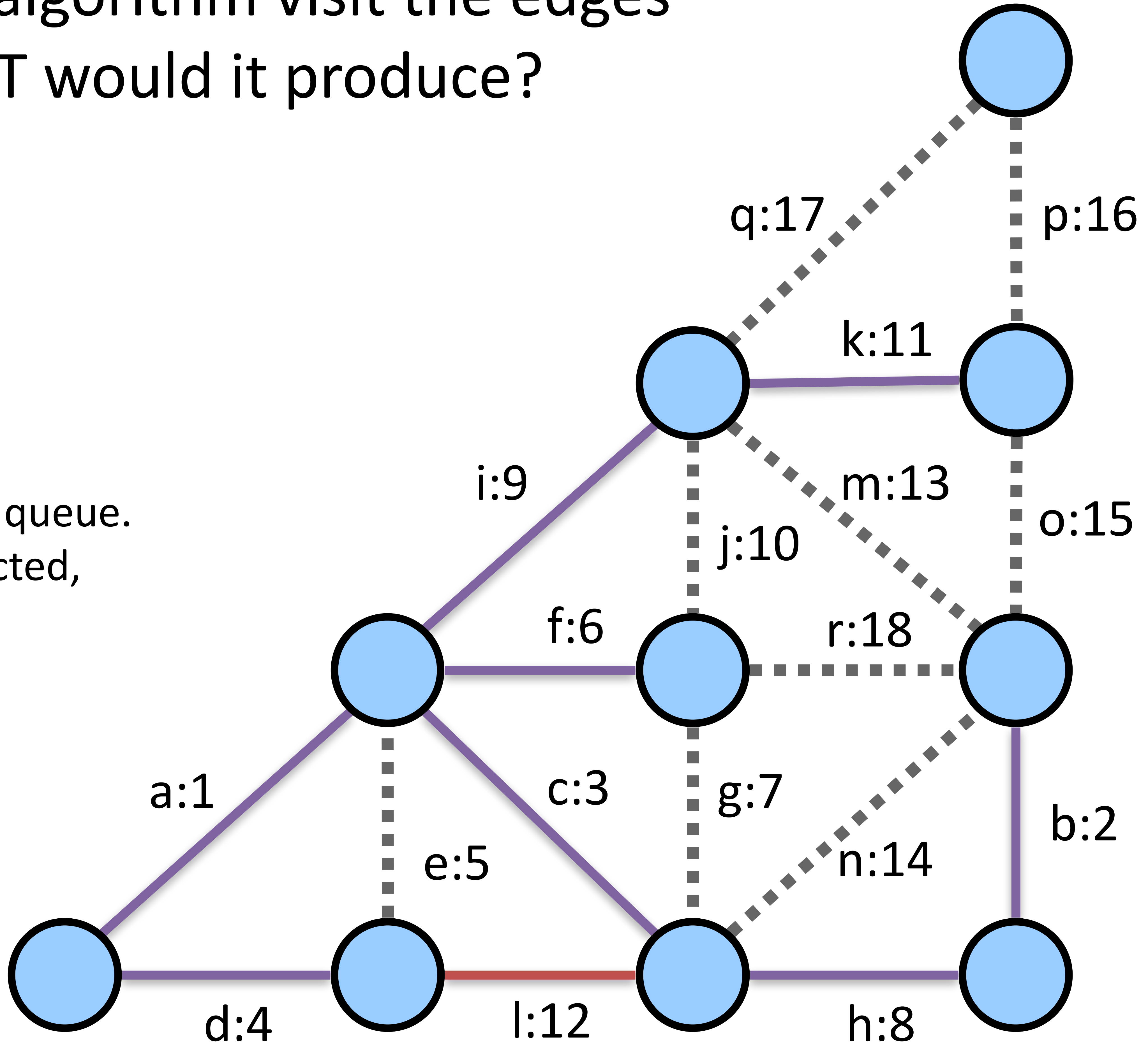


pq = {**k:11**, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
          Dequeue an edge e from the priority queue.
          If e's endpoints aren't already connected,
             add that edge into the graph.
          Otherwise, skip the edge.
```



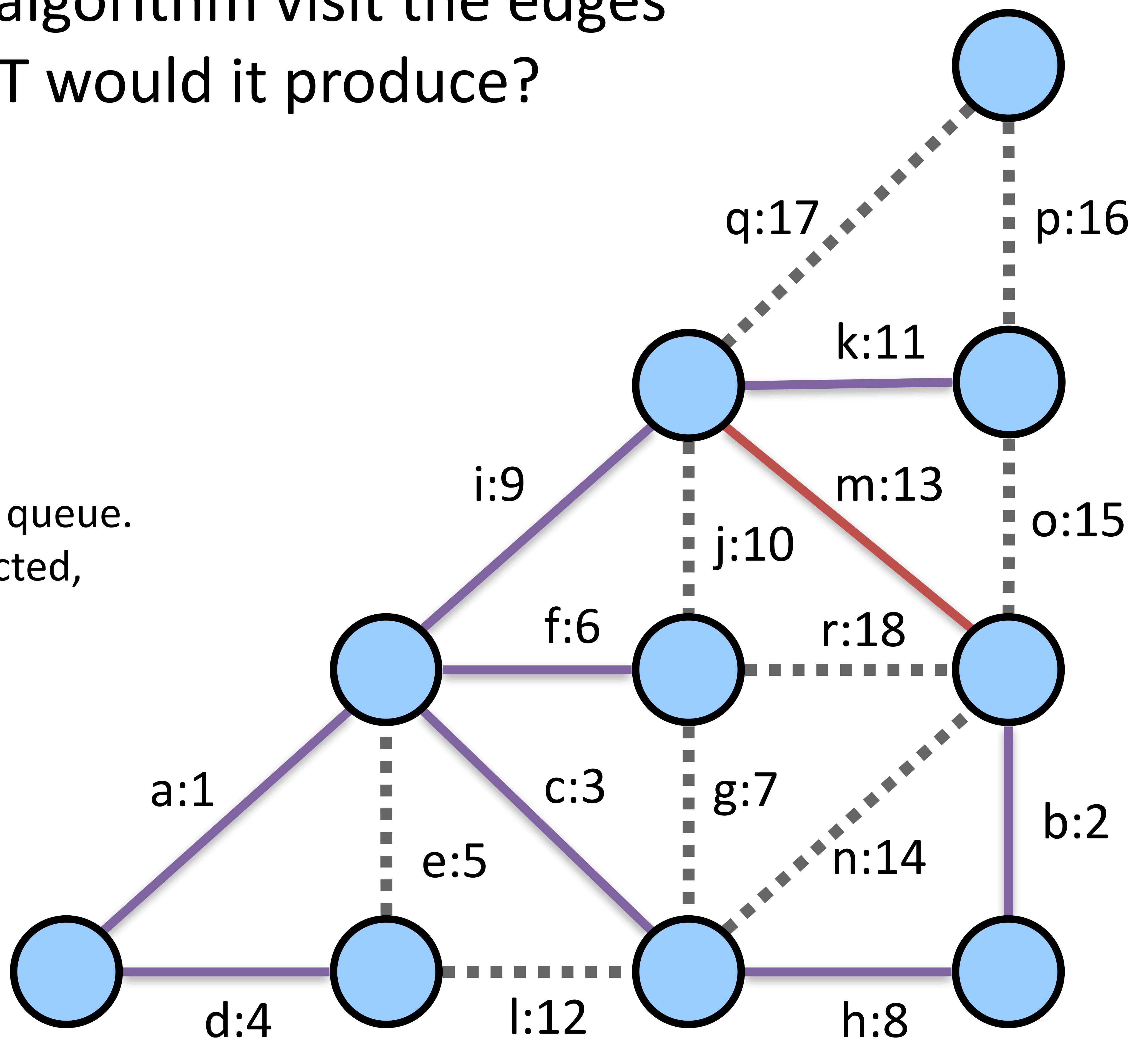
pq = {**l:12**, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
         Dequeue an edge e from the priority queue.
         If e's endpoints aren't already connected,
            add that edge into the graph.
         Otherwise, skip the edge.
```

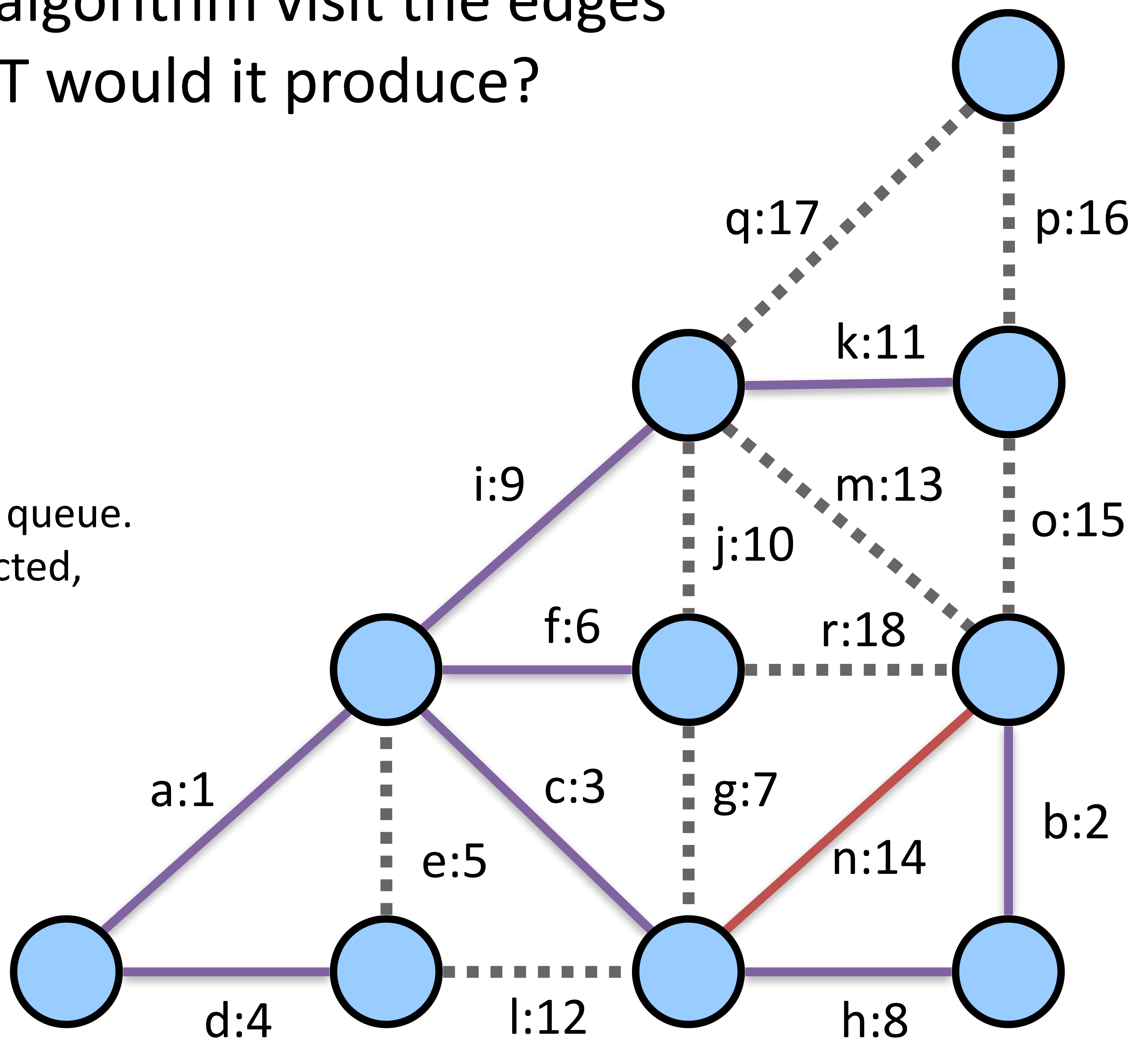
pq = {**m:13**, n:14, o:15, p:16, q:17, r:18}



Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function **kruskal**(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
             Dequeue an edge e from the priority queue.
             If e's endpoints aren't already connected,
                add that edge into the graph.
             Otherwise, skip the edge.
```

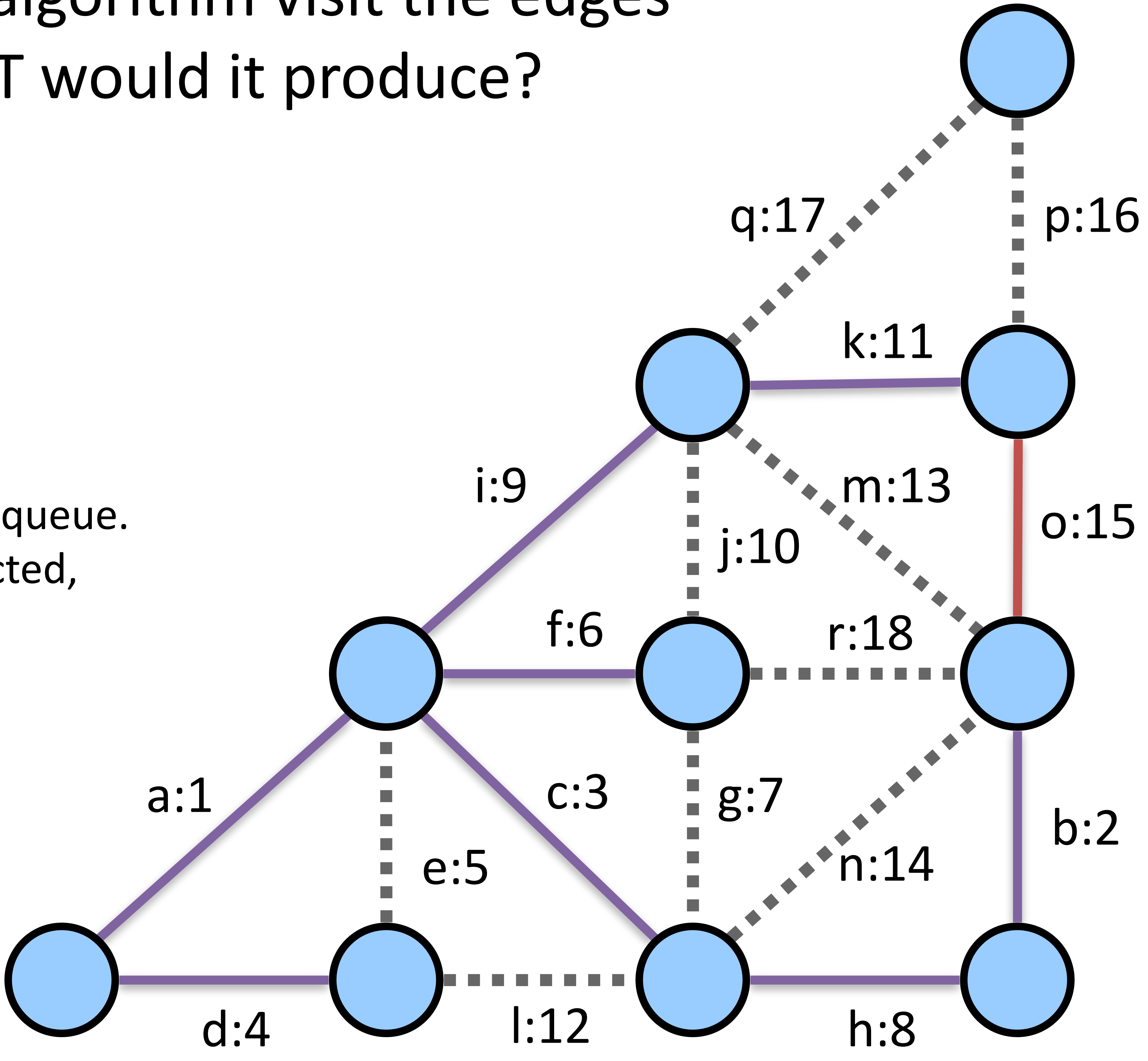


pq = {**n:14**, o:15, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
         Dequeue an edge e from the priority queue.
         If e's endpoints aren't already connected,
            add that edge into the graph.
         Otherwise, skip the edge.
```



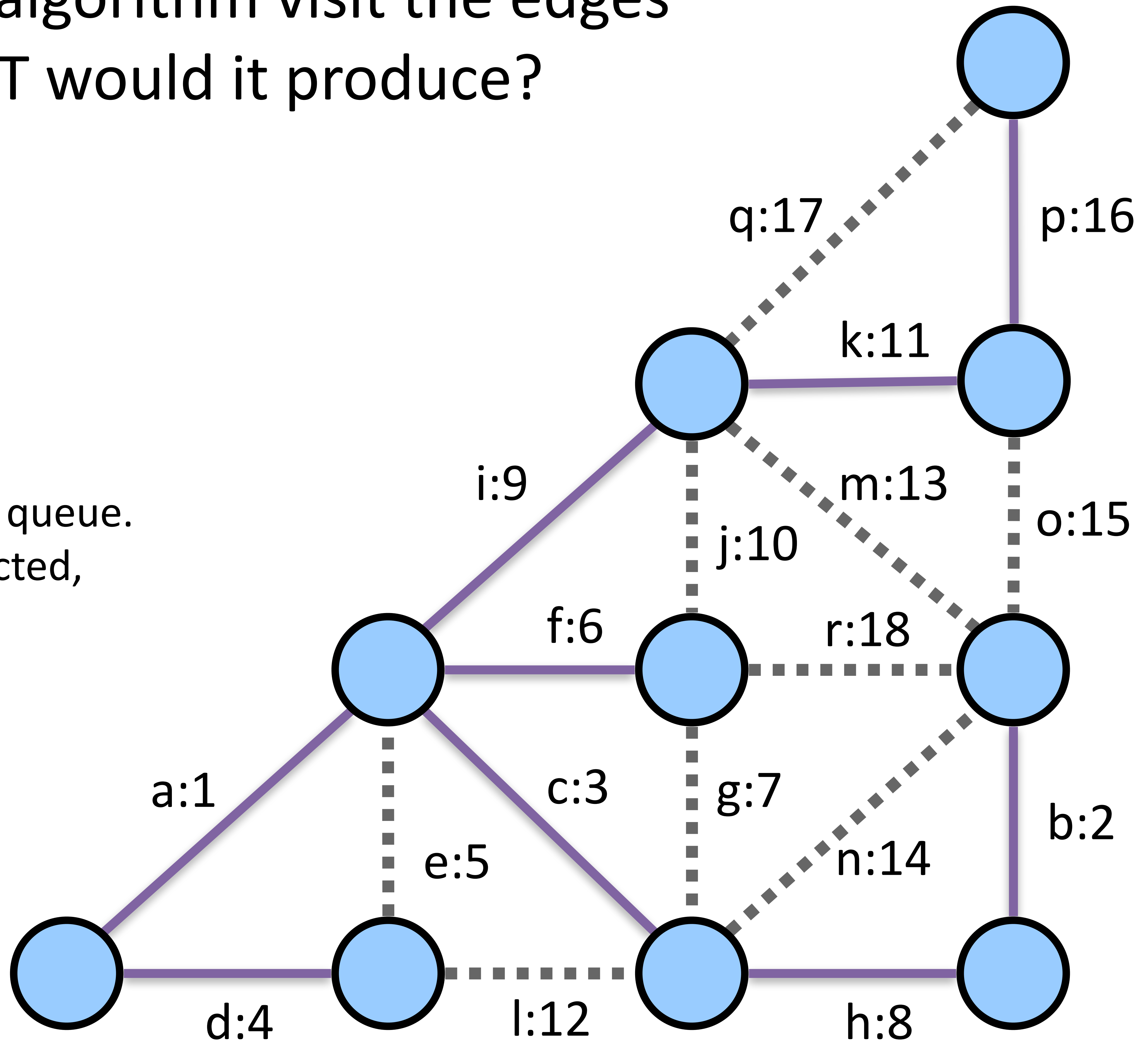
pq = {**o:15**, p:16, q:17, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
           Dequeue an edge e from the priority queue.
           If e's endpoints aren't already connected,
              add that edge into the graph.
           Otherwise, skip the edge.
```

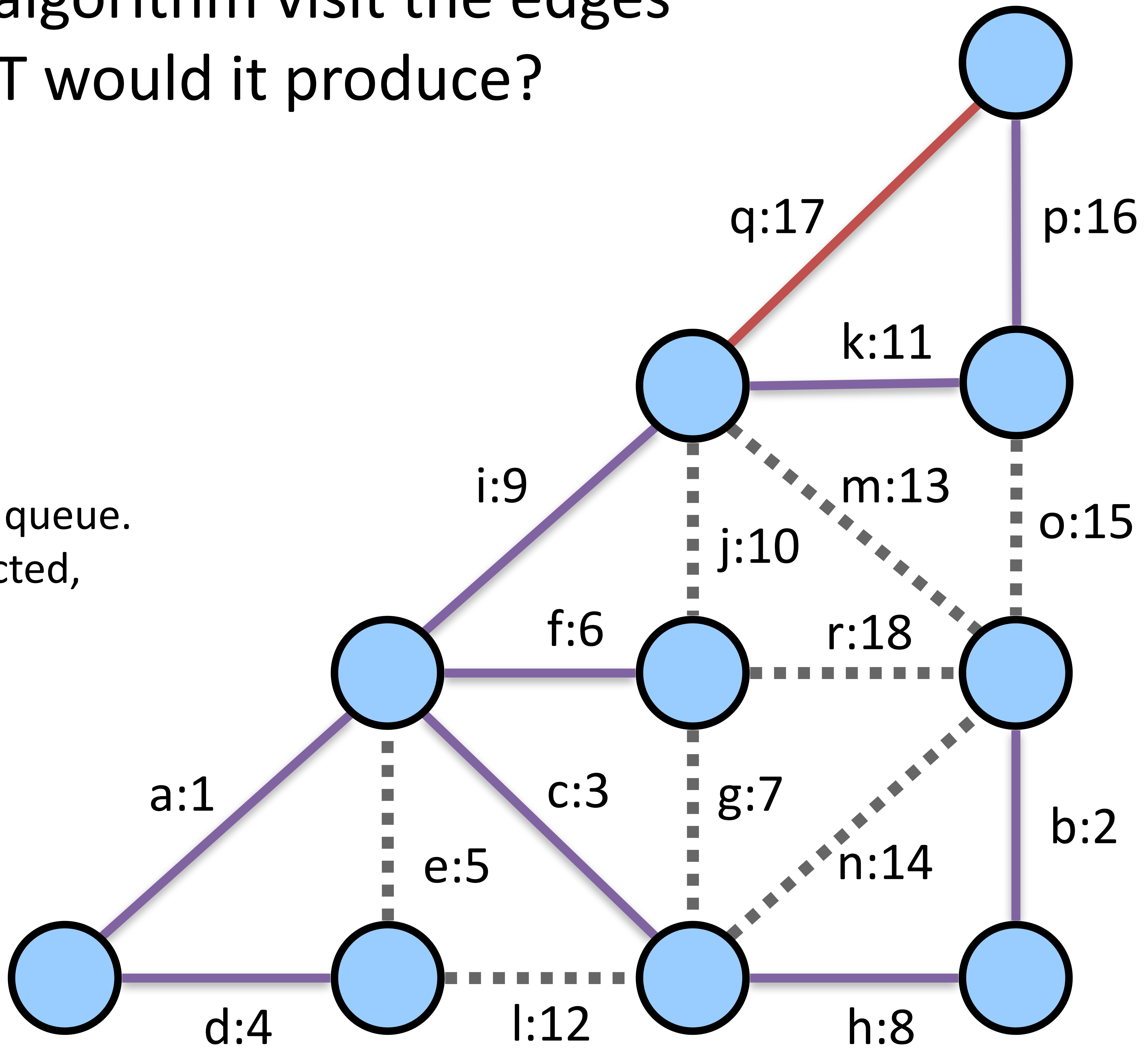
pq = {**p:16**, q:17, r:18}



Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function **kruskal**(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
           Dequeue an edge e from the priority queue.
           If e's endpoints aren't already connected,
              add that edge into the graph.
           Otherwise, skip the edge.
```



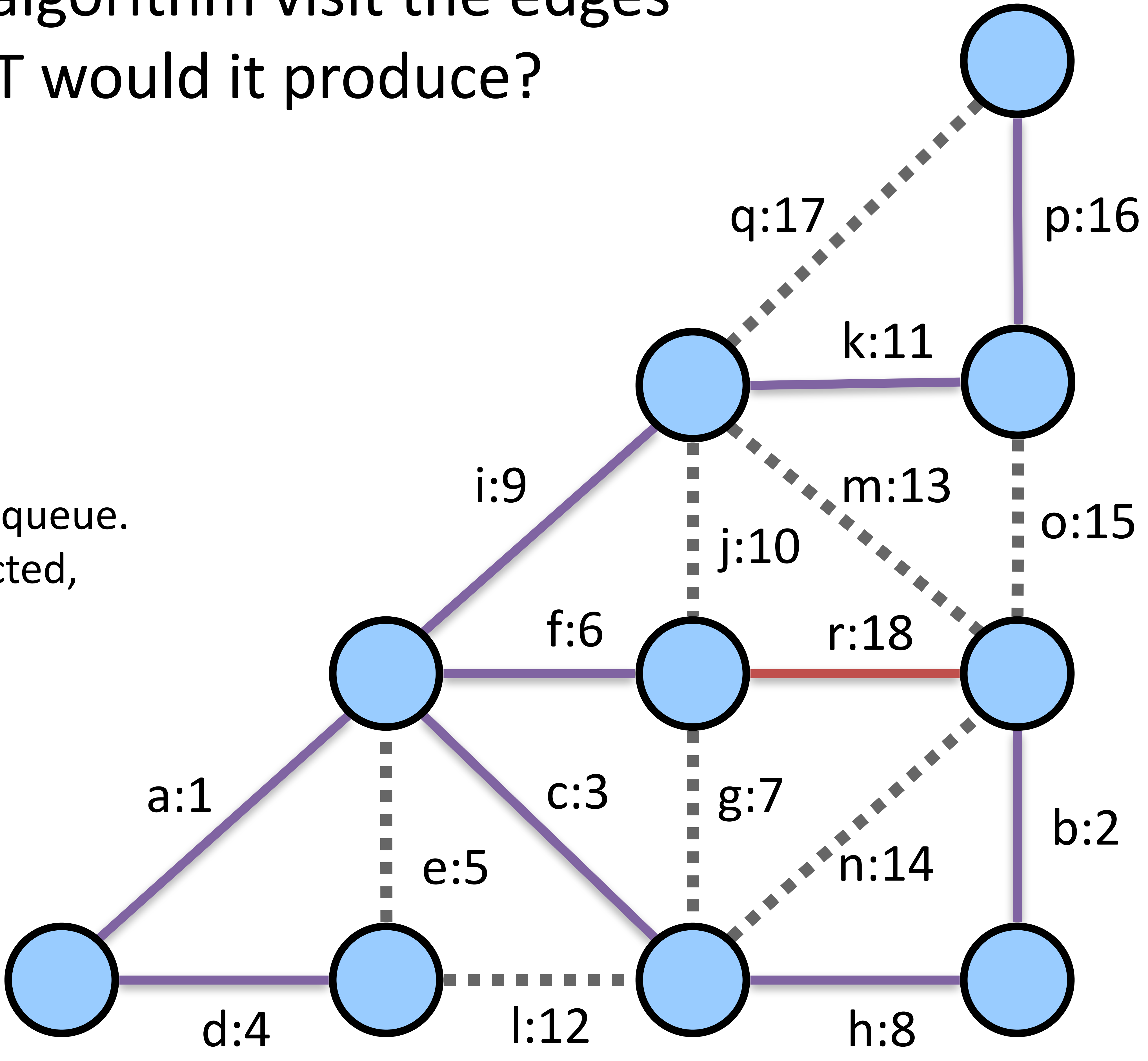
pq = {**q:17**, r:18}

Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
  Remove all edges from the graph.
  Place all edges into a priority queue
       based on their weight (cost).
  While the priority queue is not empty:
         Dequeue an edge e from the priority queue.
         If e's endpoints aren't already connected,
            add that edge into the graph.
         Otherwise, skip the edge.
```

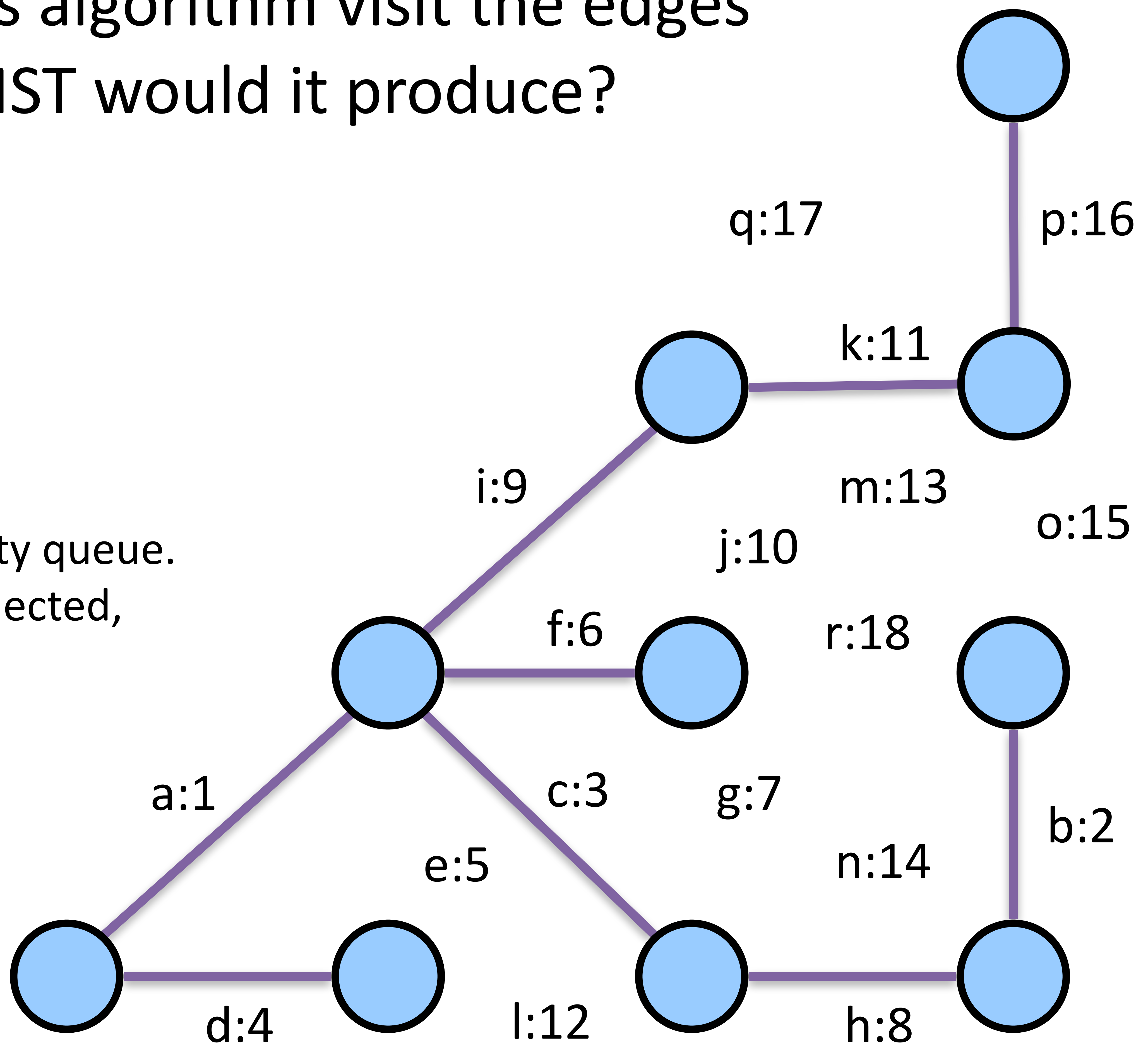
pq = {**r:18**}



Kruskal Example

- In what order would Kruskal's algorithm visit the edges in the graph below? What MST would it produce?

```
function kruskal(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
           Dequeue an edge e from the priority queue.
           If e's endpoints aren't already connected,
              add that edge into the graph.
           Otherwise, skip the edge.
```



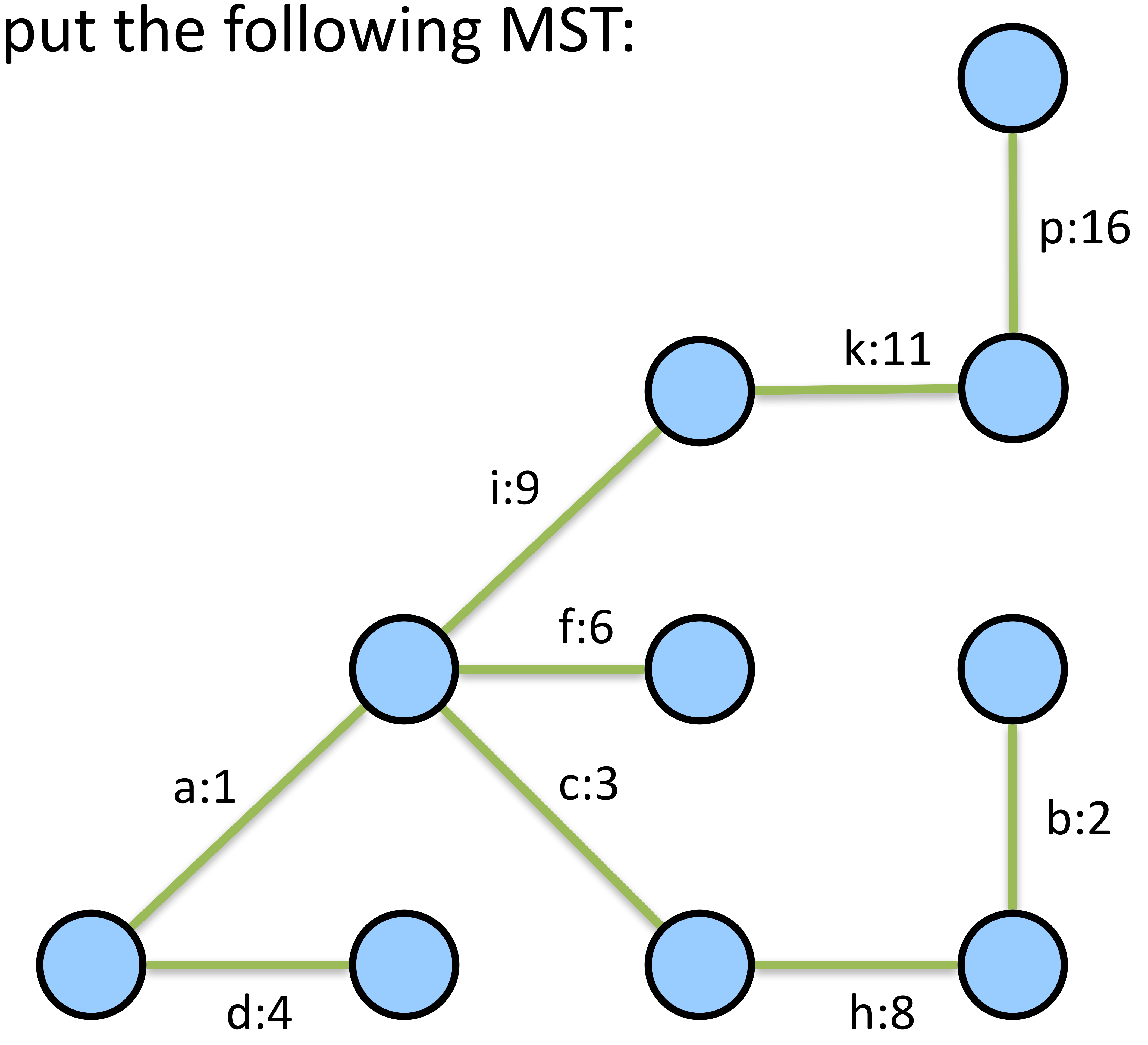
pq = {}

Kruskal Example

- Kruskal's algorithm would output the following MST:
  - {a, b, c, d, f, h, i, k, p}

- The MST's total cost is:

  1+2+3+4+6+8+9+11+16 = 60

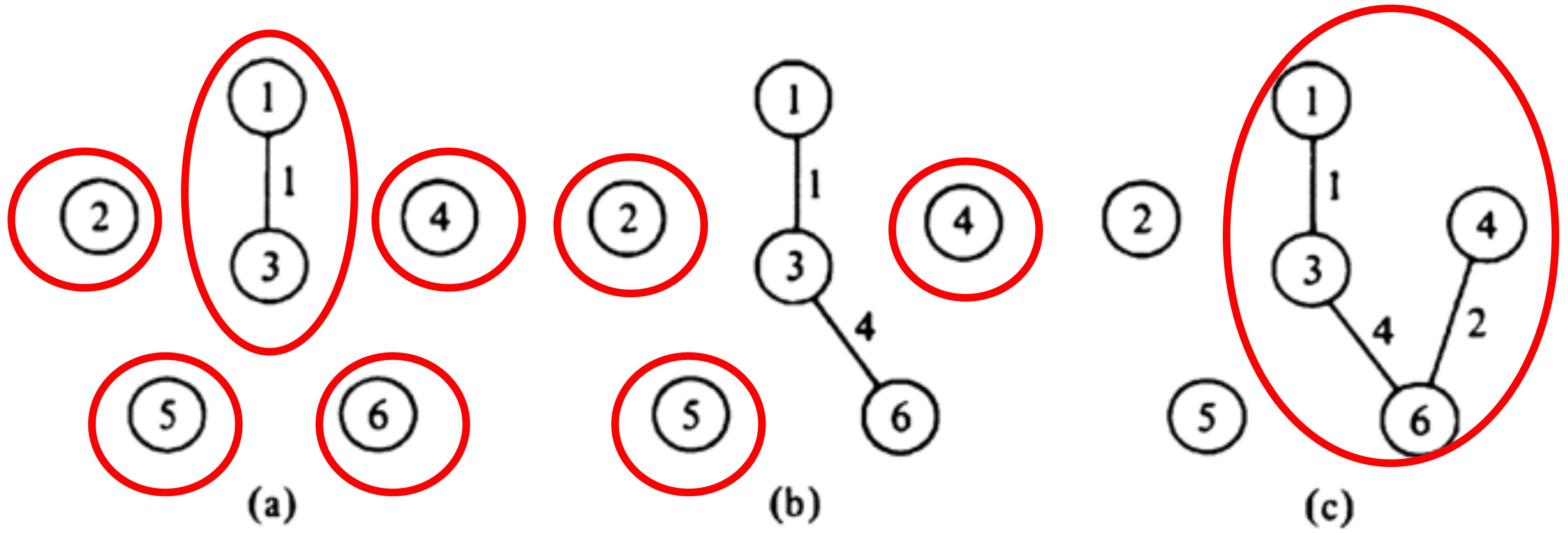


- What data structures should we use to implement this algorithm?

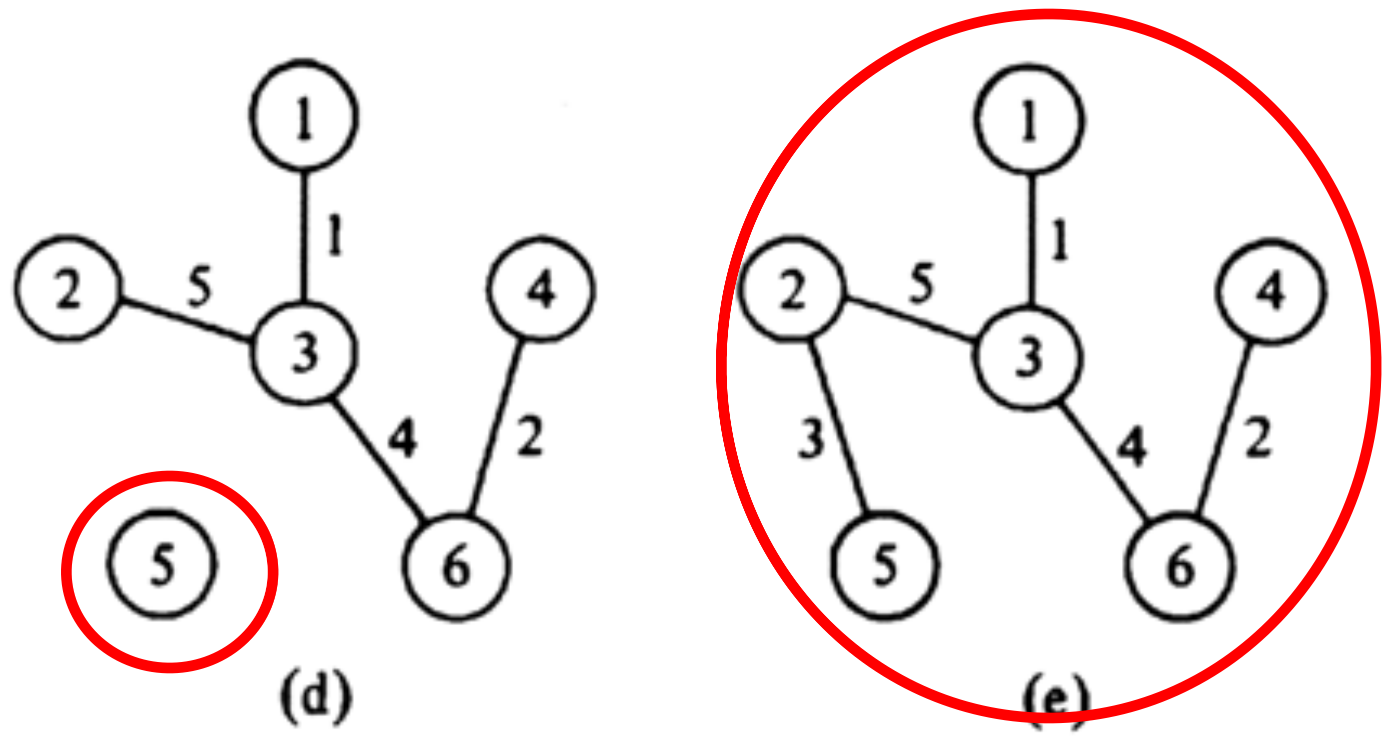
```
function **kruskal**(graph):
 Remove all edges from the graph.
 Place all edges into a priority queue
       based on their weight (cost).
 While the priority queue is not empty:
           Dequeue an edge e from the priority queue.
           If e's endpoints aren't already connected,
              add that edge into the graph.
           Otherwise, skip the edge.
```

- Need some way to identify which vertexes are "connected" to which other ones
  - we call these "**clusters**" of vertices

- Also need an efficient way to figure out which cluster a given vertex is in.



- Also need to **merge clusters** when adding an edge.



References and Advanced Reading

- **References:**
  - Minimum Spanning Tree visualization: https://visualgo.net/mst
  - Kruskal's Algorithm: https://en.wikipedia.org/wiki/Kruskal's_algorithm

- **Advanced Reading:**
  - How Internet Routing works: https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm
  - http://www.explainthatstuff.com/internet.html