CS 106B, Lecture 24
Dijkstra’s and Kruskal’s

Plan for Today

- Real-world graph algorithms (with coding examples!)
  - **Dijkstra's Algorithm** for finding the **least-cost path** (like Google Maps)
  - **Kruskal's Algorithm** for finding the **minimum spanning tree**
    - Applications in civil engineering and biology

Shortest Paths

- Recall: BFS allows us to find the shortest path
- Sometimes, you want to find the **least-cost path**
  - Only applies to graphs with **weighted** edges
- Examples:
  - cheapest flight(s) from here to New York
  - fastest driving route (Google Maps)
  - the internet: fastest path to send information through the network of routers

Least-Cost Paths

- BFS uses a **queue** to keep track of which nodes to use next
- BFS pseudocode:

```
bfs from v1:
    add v1 to the queue.
    while queue is not empty:
      dequeue a node n
      enqueue n's unseen neighbors
```

- How could we modify this pseudocode to dequeue the **least-cost** nodes instead of the **closest nodes**?

Least-Cost Paths

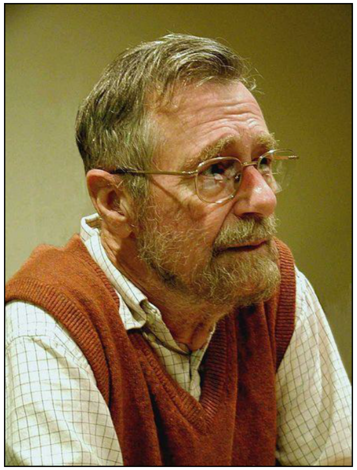
- BFS uses a **queue** to keep track of which nodes to use next
- BFS pseudocode:

```
bfs from v1:
    add v1 to the queue.
    while queue is not empty:
      dequeue a node n
      enqueue n's unseen neighbors
```

- How could we modify this pseudocode to dequeue the **least-cost** nodes instead of the **closest nodes**?
  - Use a **priority queue** instead of a queue

Edsger Dijkstra (1930-2002)

- famous Dutch computer scientist and prof. at UT Austin
  - Turing Award winner (1972)



- Noteworthy algorithms and software:
  - THE multiprogramming system (OS)
    - layers of abstraction
  - Compiler for a language that can do recursion
  - Dijkstra's algorithm
  - Dining Philosophers Problem: resource contention, deadlock
- famous papers:
  - "Go To considered harmful"
  - "On the cruelty of really teaching computer science"

Dijkstra pseudocode

dijkstra(v_1, v_2):

consider every vertex to have a cost of infinity, except v_1 which has a cost of 0.
create a *priority queue* of vertexes, ordered by cost, storing only v_1 .

while the *pqueue* is not empty:
dequeue a vertex *v* from the *pqueue*, and mark it as **visited**.
for each unvisited neighbor, *n*, of *v*, we can reach *n*
with a total **cost** of (*v*'s cost + the weight of the edge from *v* to *n*).
if this cost is cheaper than *n*'s current cost,
we should **enqueue** the neighbor *n* to the pqueue with this new cost,
and remember *v* was its previous vertex.

when we are done, we can **reconstruct the path** from v_2 back to v_1
by following the path of previous vertices.

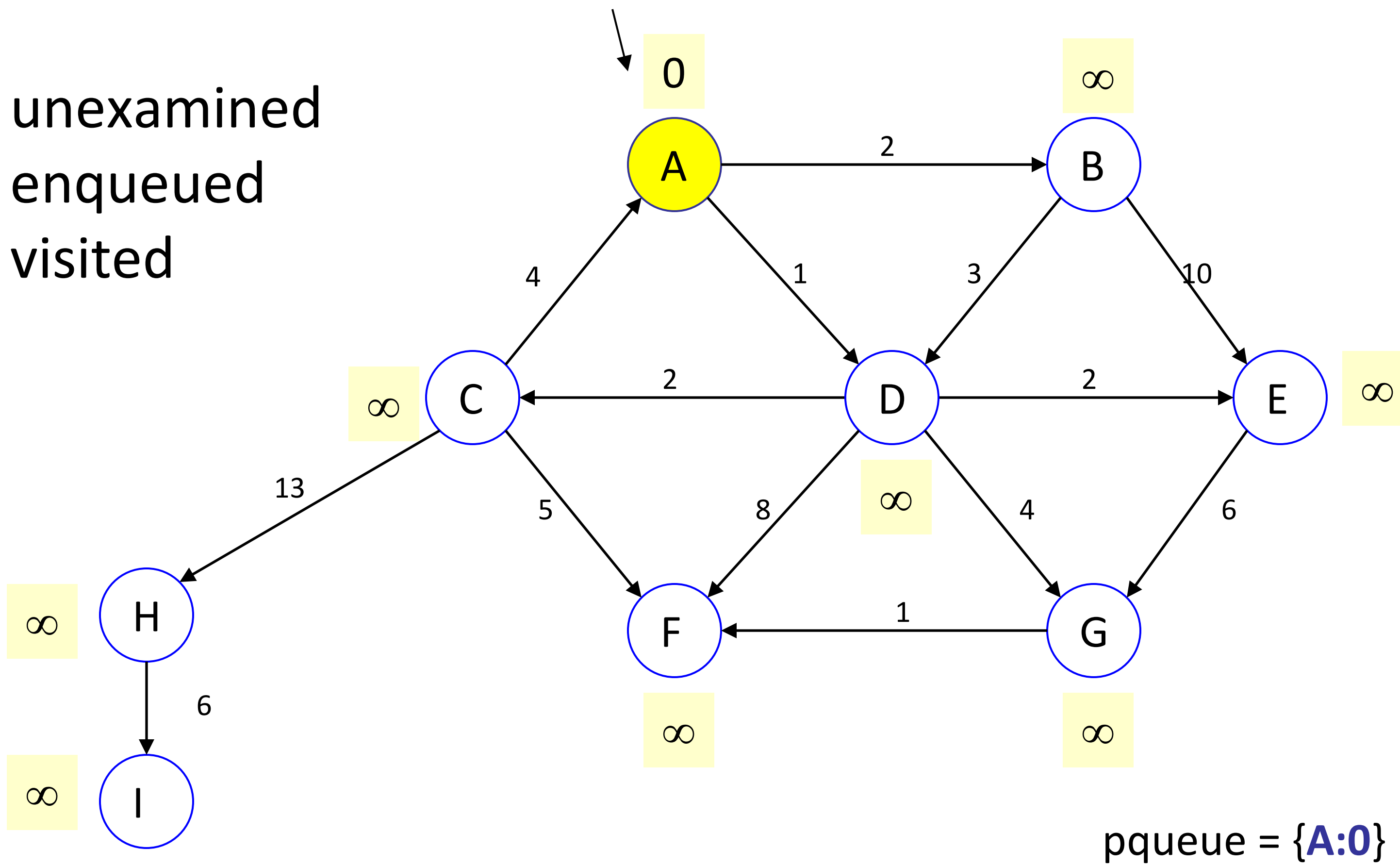
Dijkstra example

dijkstra(A, F);

- color key
  - – **white**: unexamined
  - – **yellow**: enqueued
  - – **green**: visited

v_1's distance := 0.

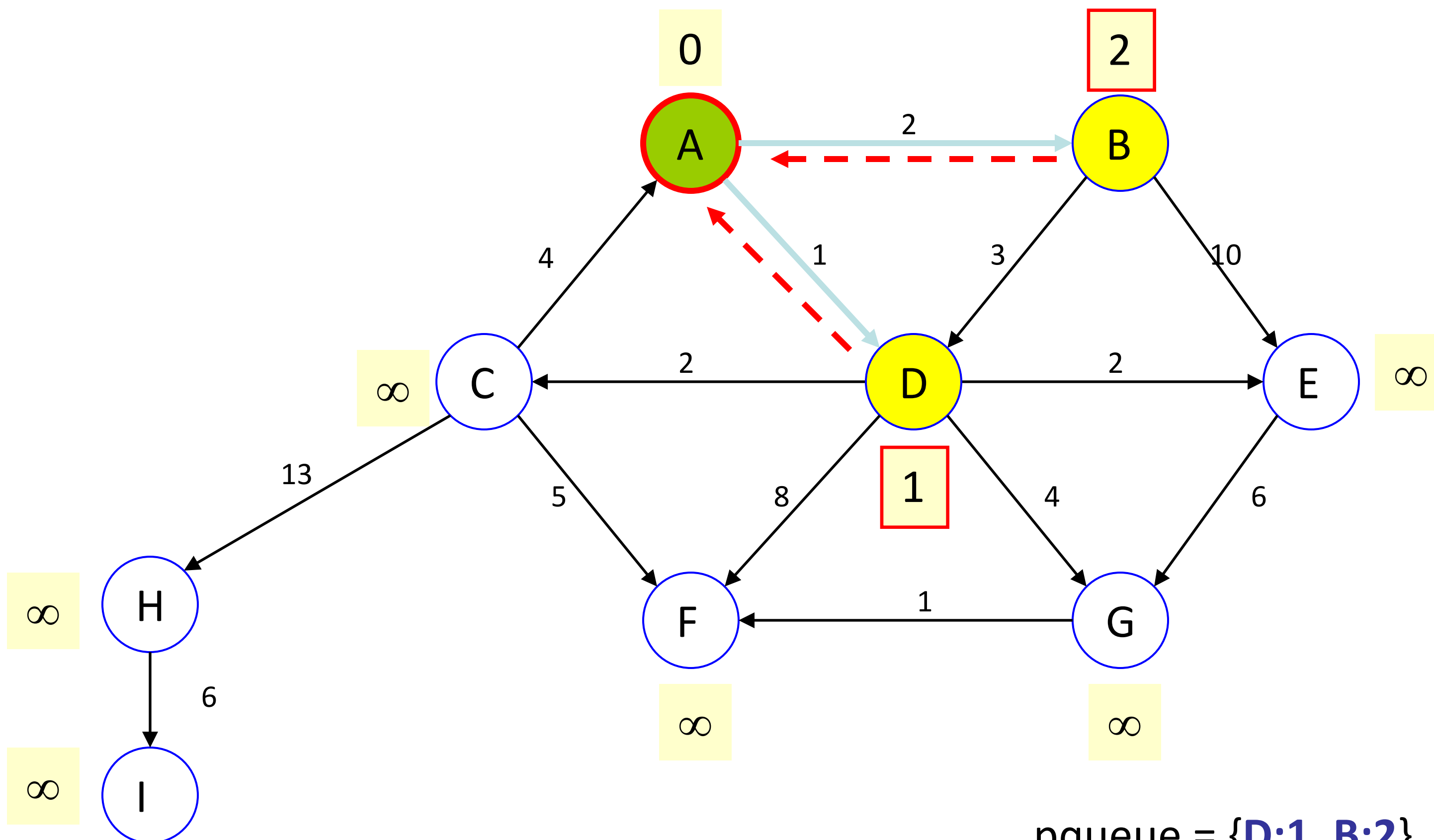
all other distances := ∞.



pqueue = {**A:0**}

Dijkstra example

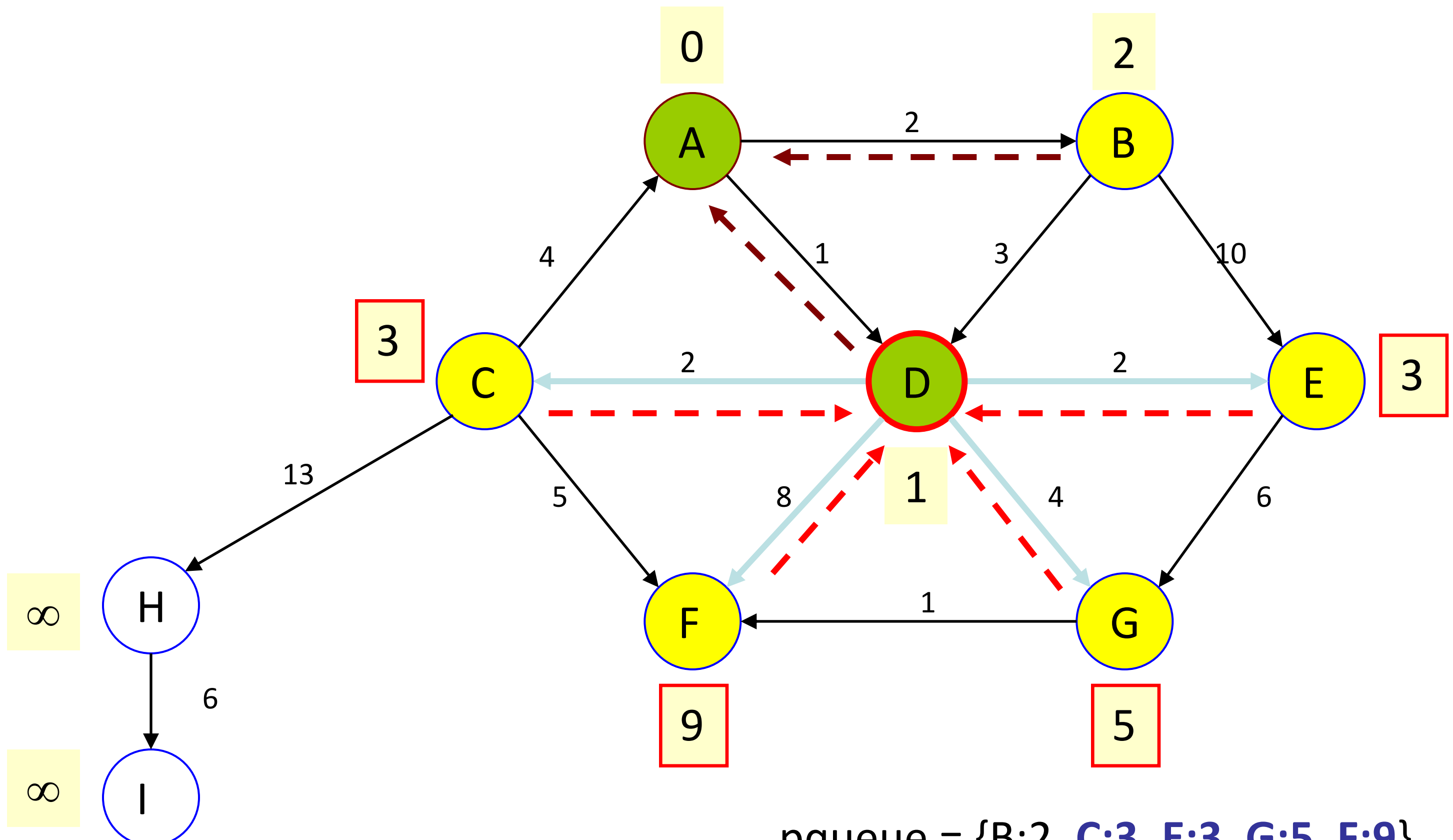
dijkstra(A, F);



pqueue = {**D:1**, **B:2**}

Dijkstra example

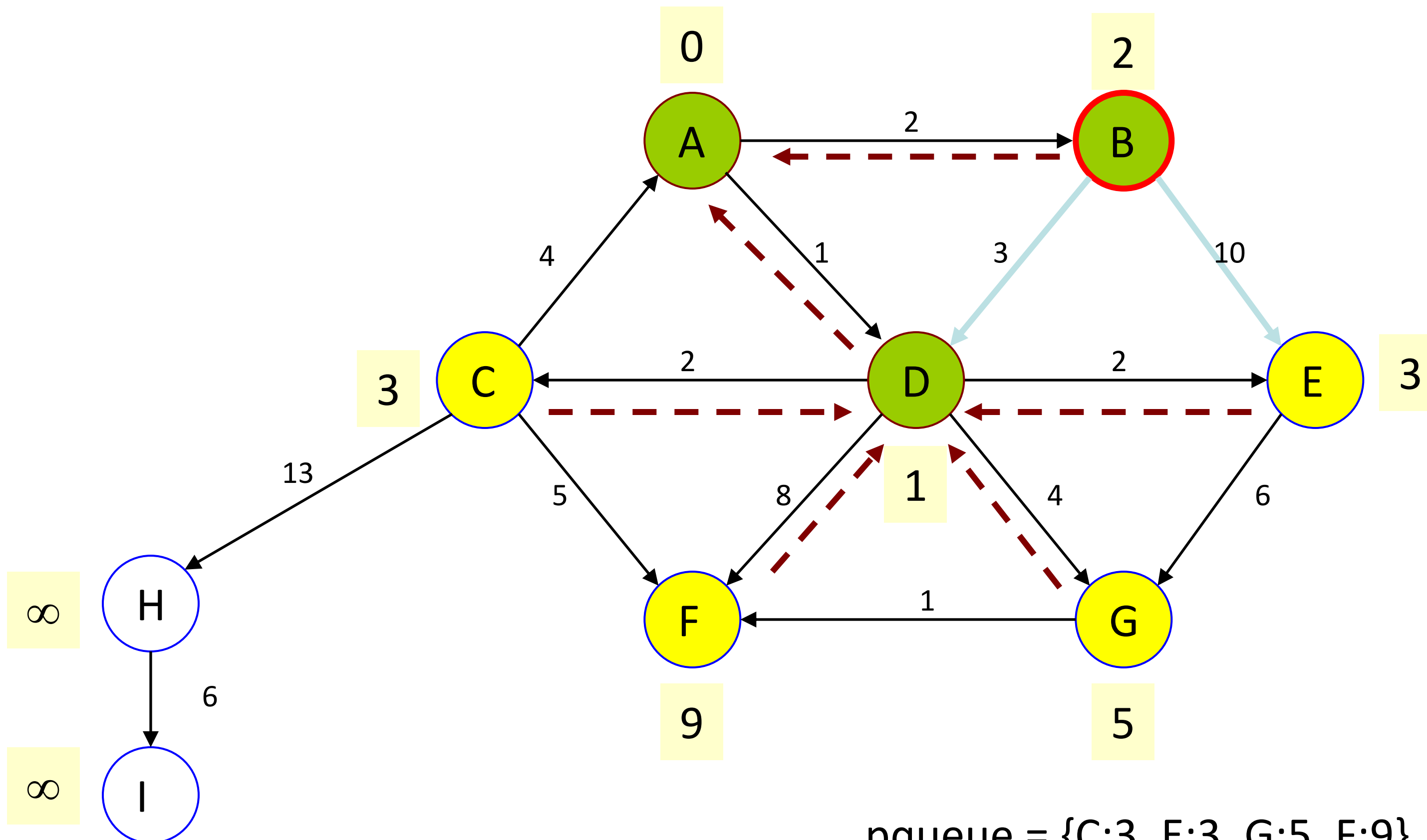
dijkstra(A, F);



pqueue = {B:2, **C:3**, **E:3**, **G:5**, **F:9**}

Dijkstra example

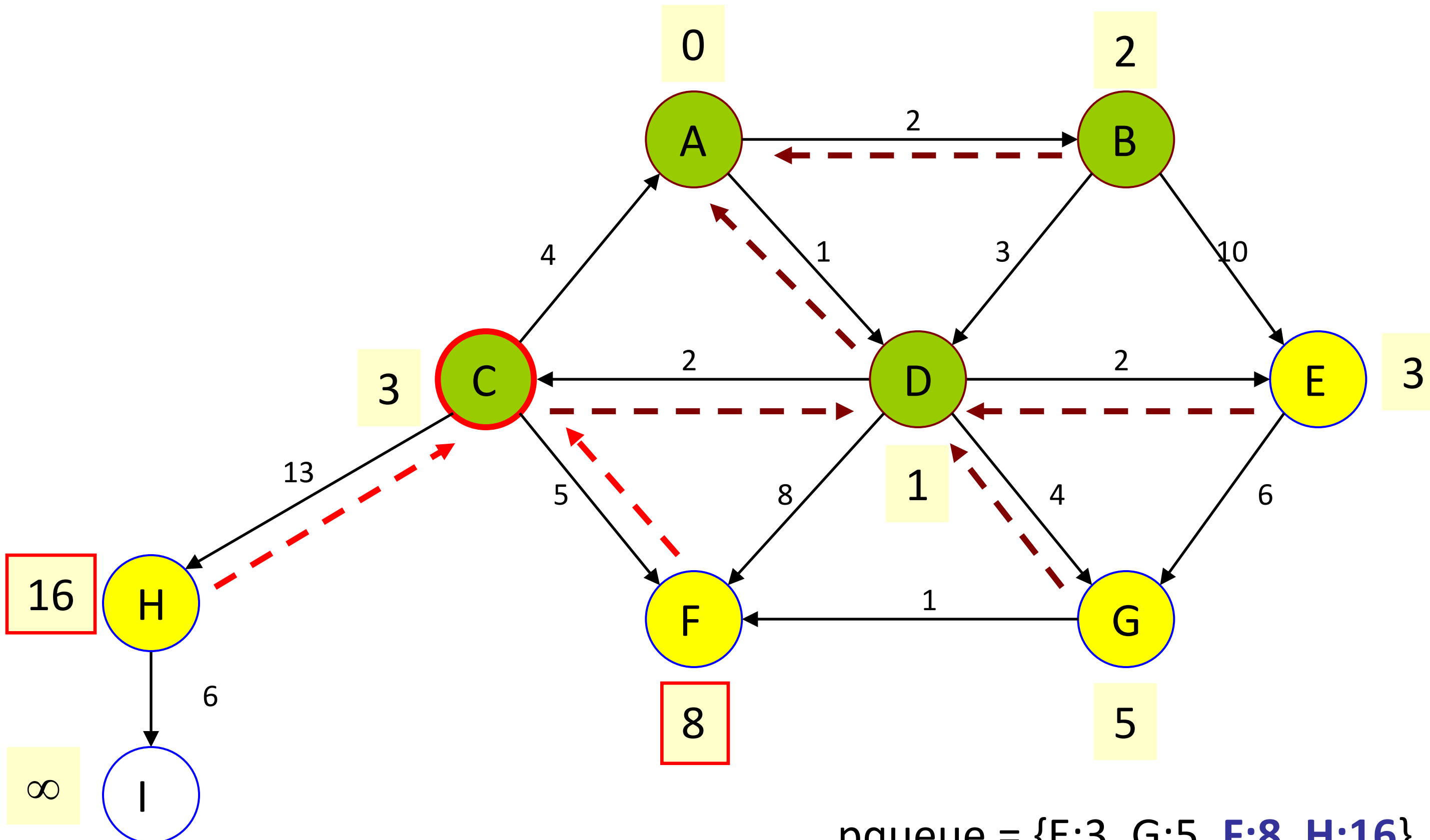
dijkstra(A, F);



pqueue = {C:3, E:3, G:5, F:9}

Dijkstra example

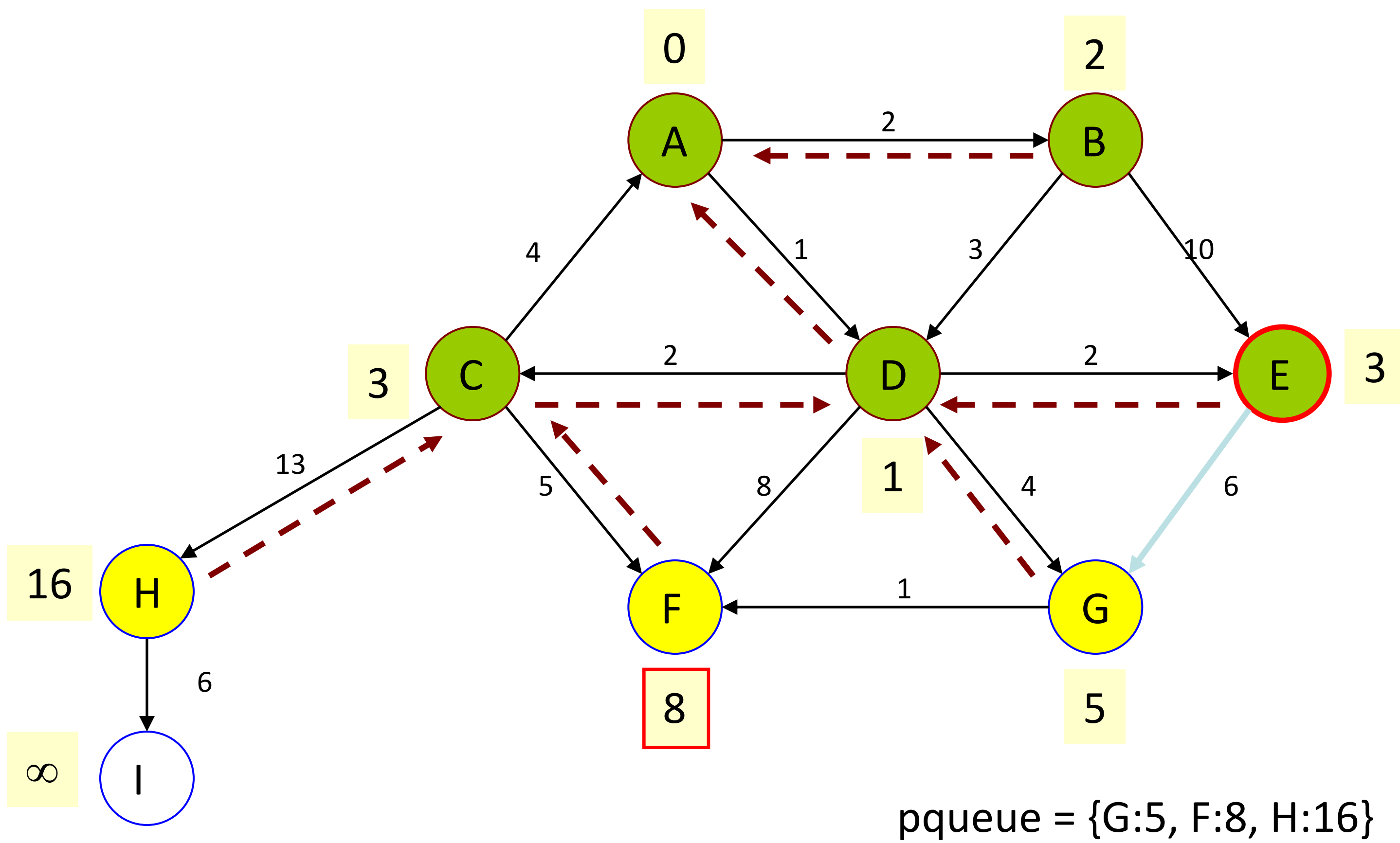
dijkstra(A, F);



pqueue = {E:3, G:5, **F:8, H:16**}

Dijkstra example

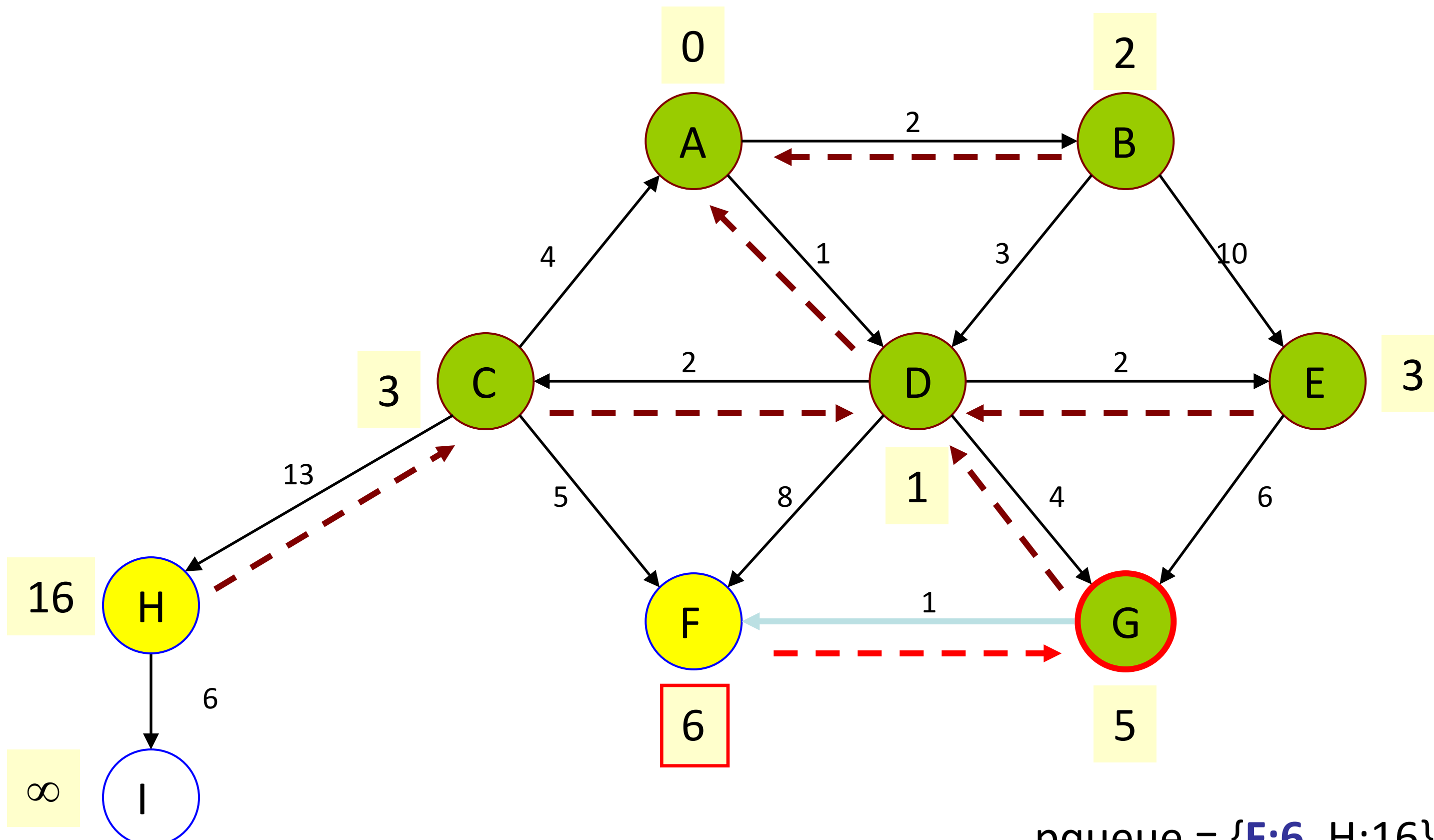
dijkstra(A, F);



pqueue = {G:5, F:8, H:16}

Dijkstra example

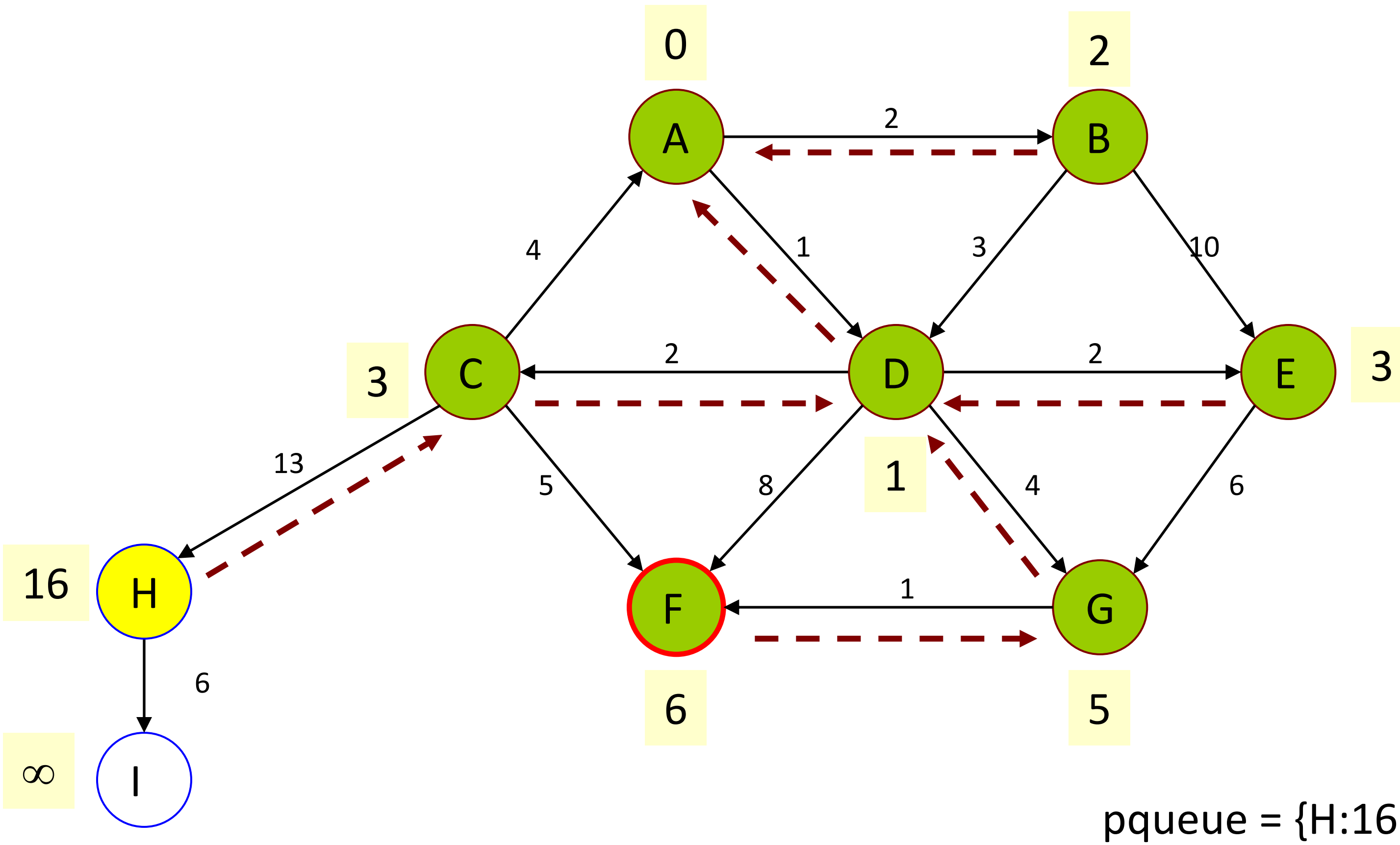
dijkstra(A, F);



pqueue = {**F:6**, H:16}

Dijkstra example

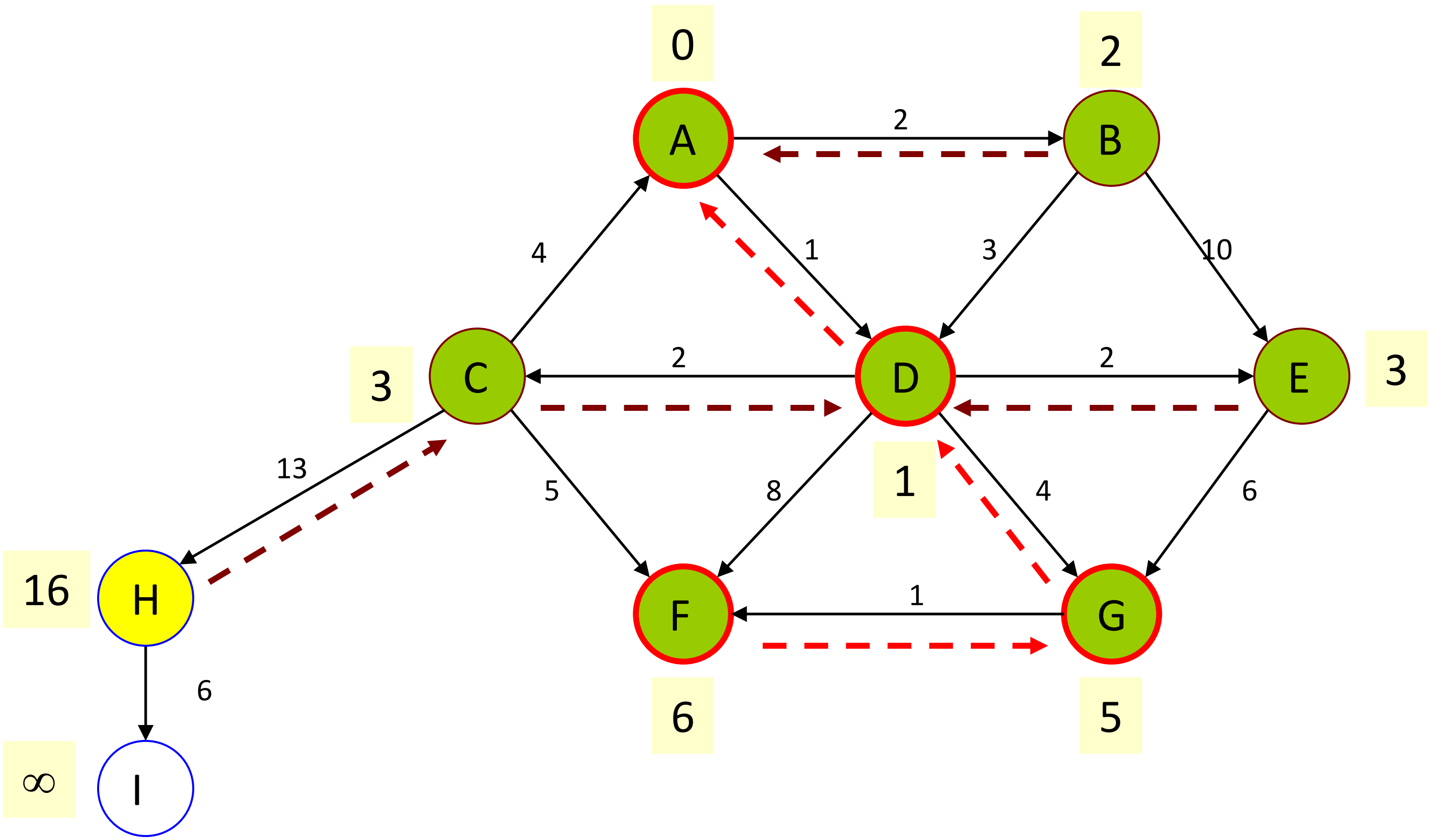
dijkstra(A, F);



pqueue = {H:16}

Dijkstra example

dijkstra(A, F);



Algorithm properties

- Dijkstra's algorithm is a *greedy algorithm*:
  - Make choices that currently seem best

- It is correct because it maintains the following two properties:
  - 1) for every marked vertex, the current recorded cost is the lowest cost to that vertex from the source vertex.
  - 2) for every unmarked vertex *v*, its recorded distance is shortest path distance to *v* from source vertex, considering only currently known vertices and *v*.

Dijkstra's runtime

- For sparse graphs, (i.e. graphs with much less than V^2 edges) Dijkstra's is implemented most efficiently with a priority queue.

  - initialization: O(*V*)
  - while loop: O(*V*) times
    - remove min-cost vertex from *pq*: O(log *V*)
    - potentially perform *E* updates on cost/previous
    - update costs in *pq*: O(log *V*)
  - reconstruct path: O(*E*)

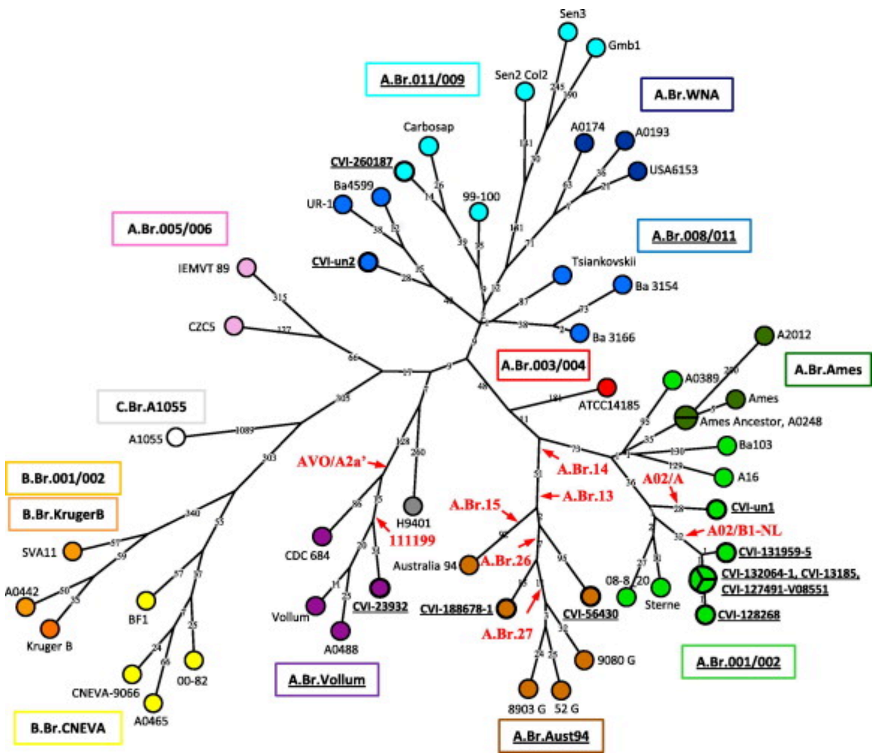
  - Total runtime: O(*V* log *V* + *E* log *V*)
    - For more in depth calculation: http://www-inst.eecs.berkeley.edu/~cs61bl/r//cur/graphs/dijkstra-algorithm-runtime.html?topic=lab24.topic&step=4&course=

Announcements

- Assn. 6 due today
- Assn. 7 (the last one!) comes out today

Minimum Spanning Trees

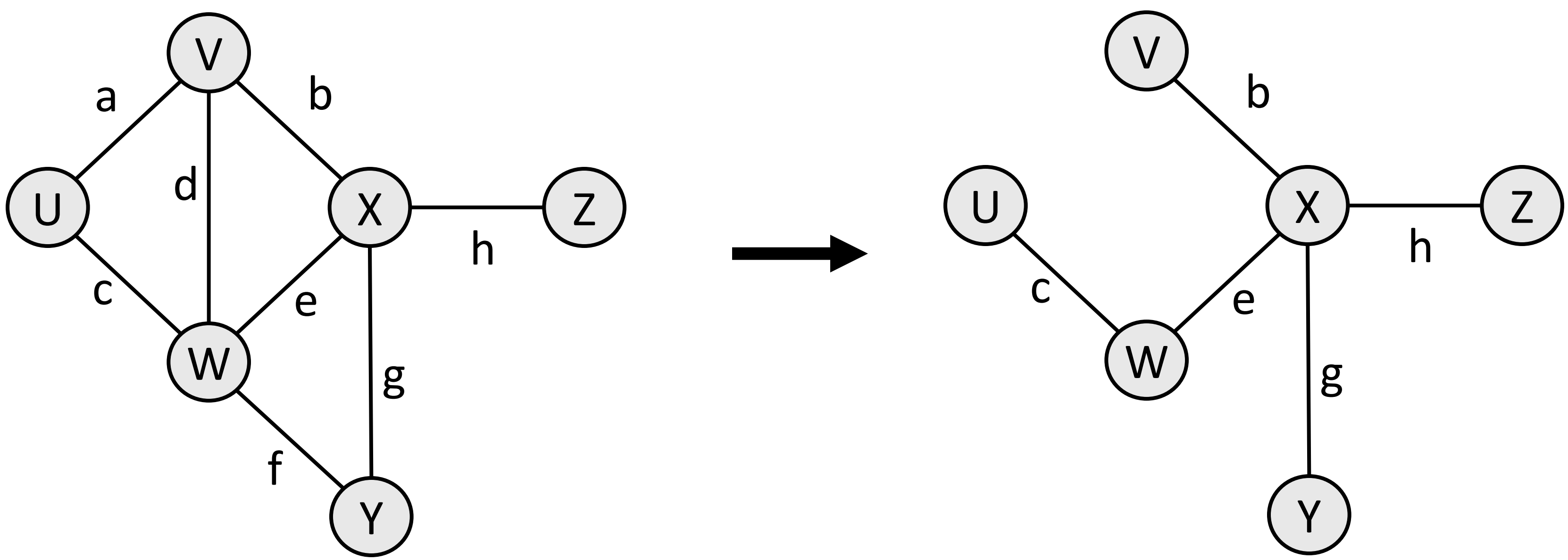
- Sometimes, you want to find a way to connect every node in a graph in the least-cost way possible
  - Utility (road, water, or power) connectivity
  - Tracing virus evolution



source: https://www.researchgate.net/figure/Position-of-the-eleven-Dutch-strains-on-the-B-anthracis-phylogenetic-tree-based-on_fig2_274406206

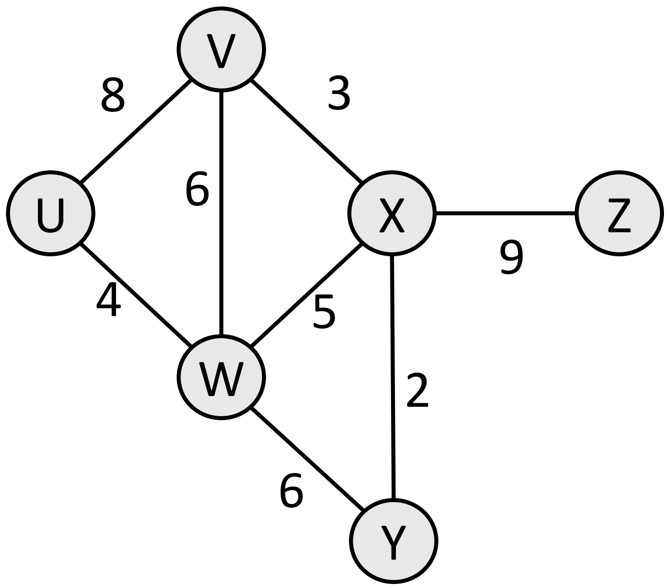
Spanning trees

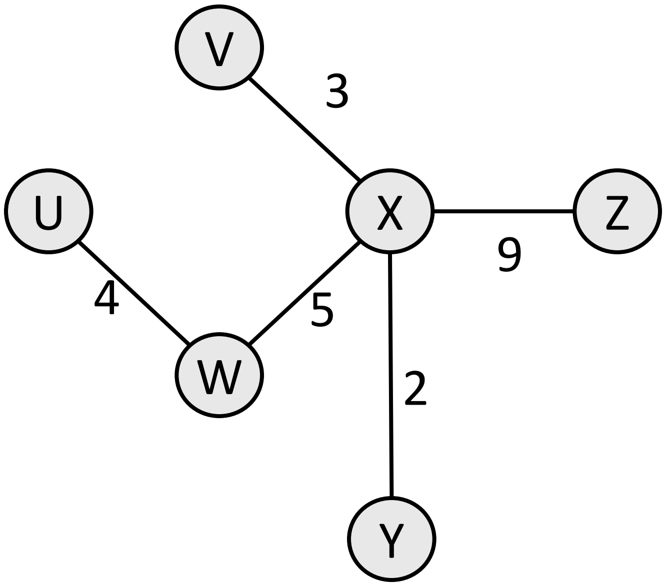
- A **spanning tree** of a graph is a set of edges that connects all vertices in the graph with no cycles.
  - What is a spanning tree for the graph below?



Minimum spanning tree

- **minimum spanning tree** (MST): A spanning tree that has the lowest combined edge weight (cost).

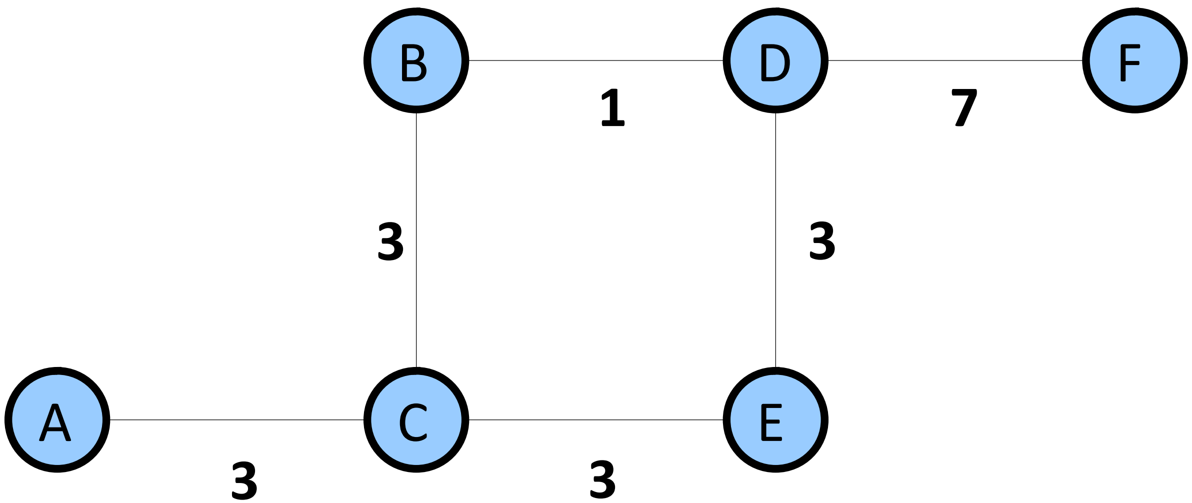




MST examples

- **Q:** How many minimum spanning trees does this graph have?

A. 0-1
B. 2-3
C. 4-5
D. 6-7
E. > 7



(question courtesy Cynthia Lee)

Kruskal's algorithm

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

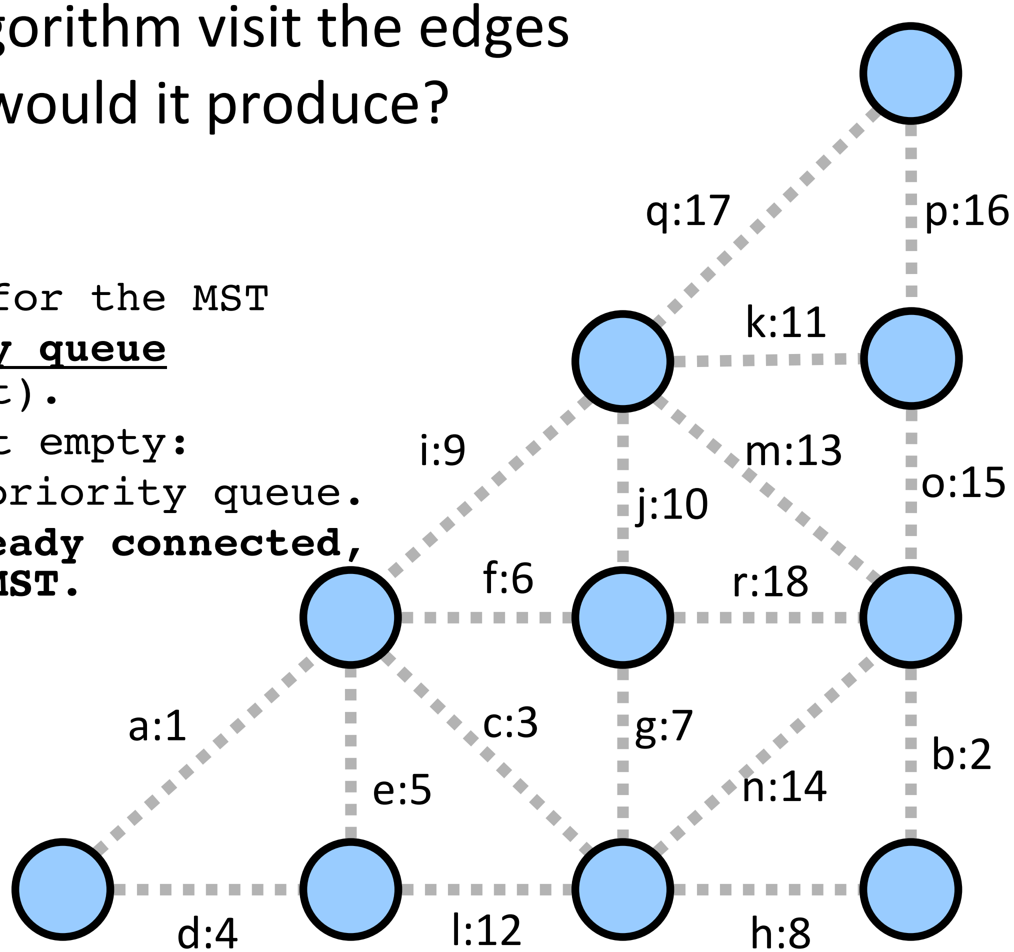
```
function kruskal(graph):
  Start with an empty structure for the MST
  Place all edges into a priority queue
      based on their weight (cost).
  While the priority queue is not empty:
    Dequeue an edge e from the priority queue.
    If e's endpoints aren't already connected,
        add that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

- **Runtime:** O(E log E) = O(E log V)

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

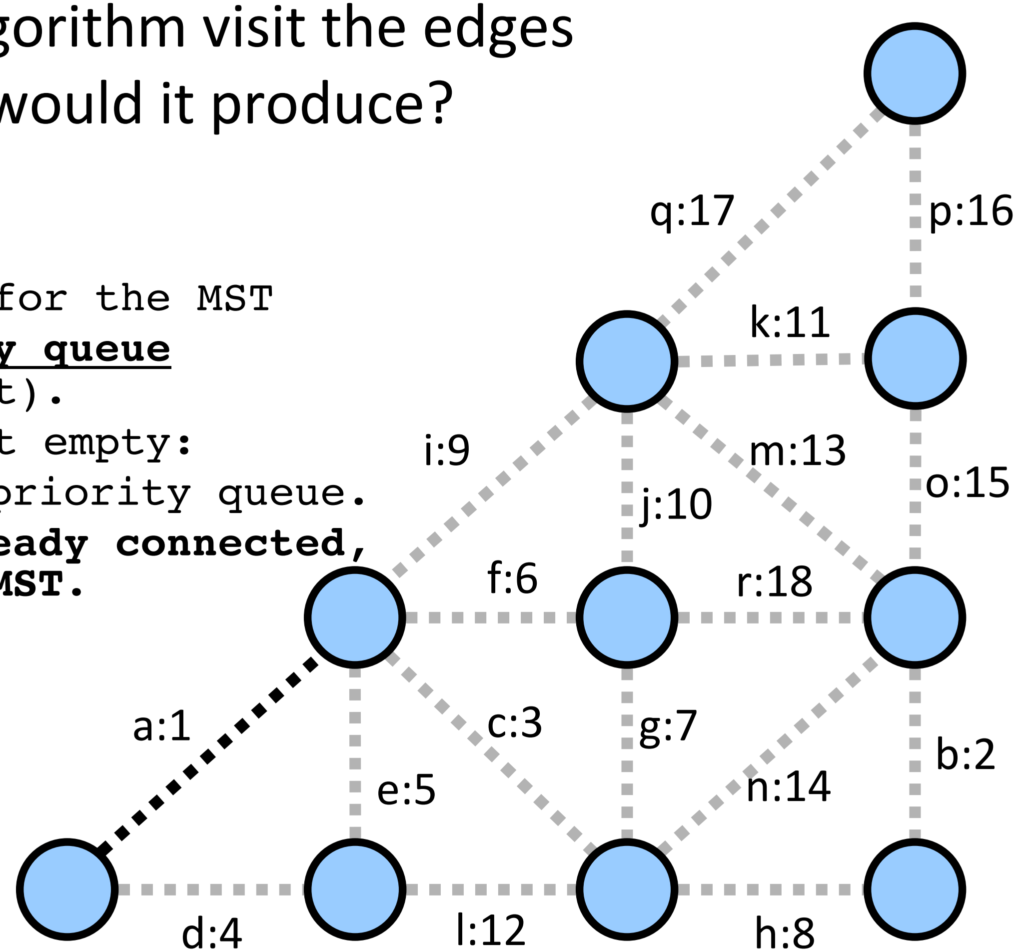


- pq = {a:1, b:2, c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

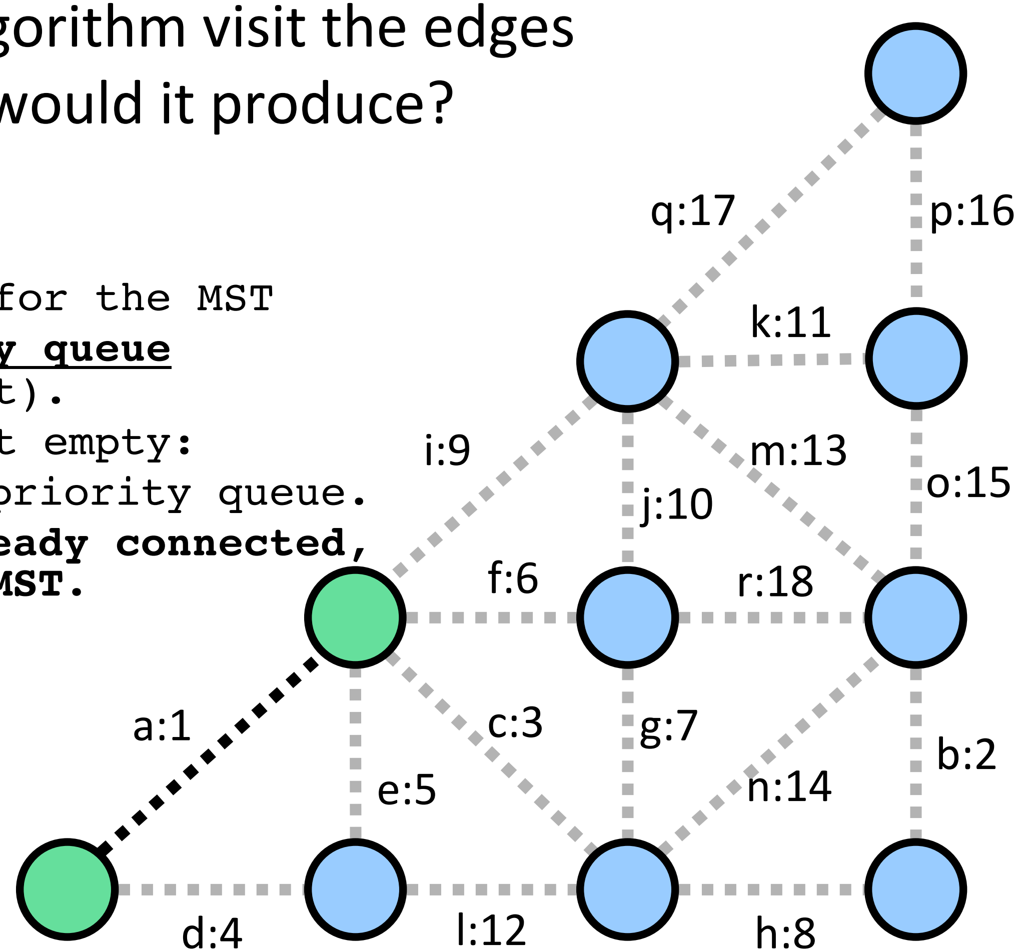


  - pq = {b:2, c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

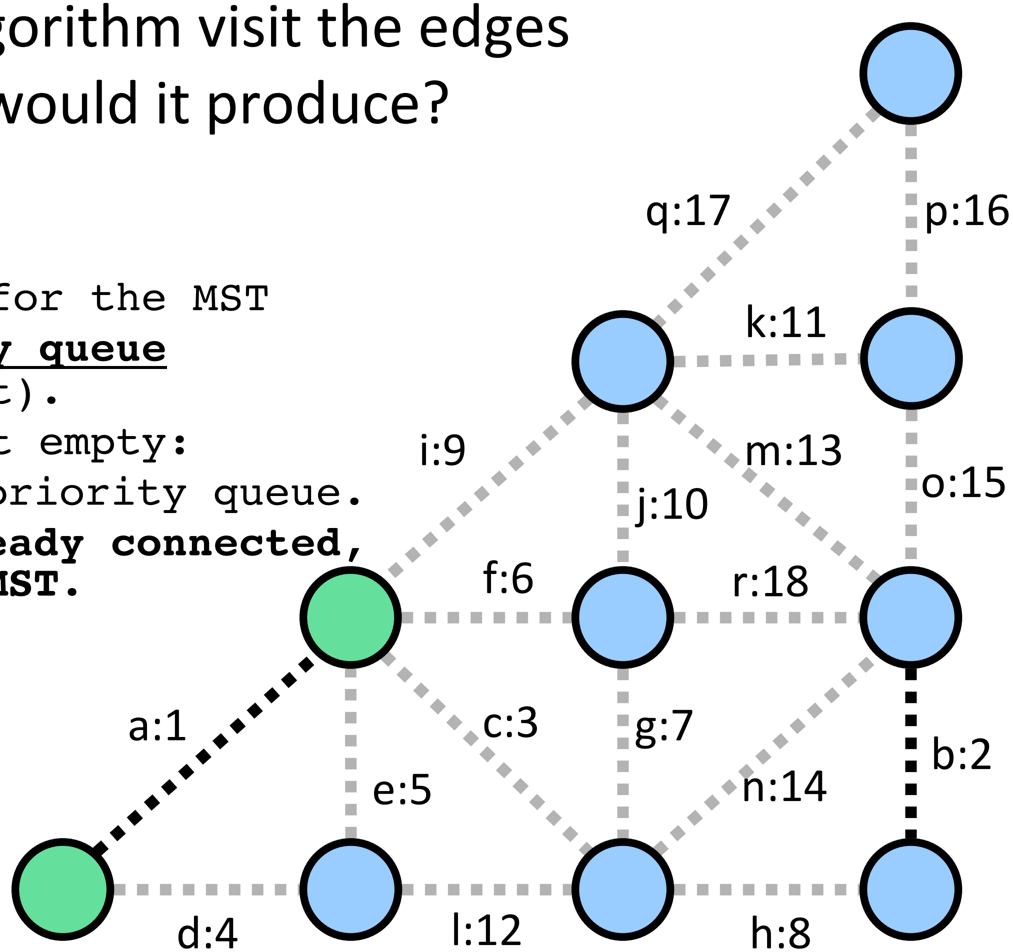


  - pq = {b:2, c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

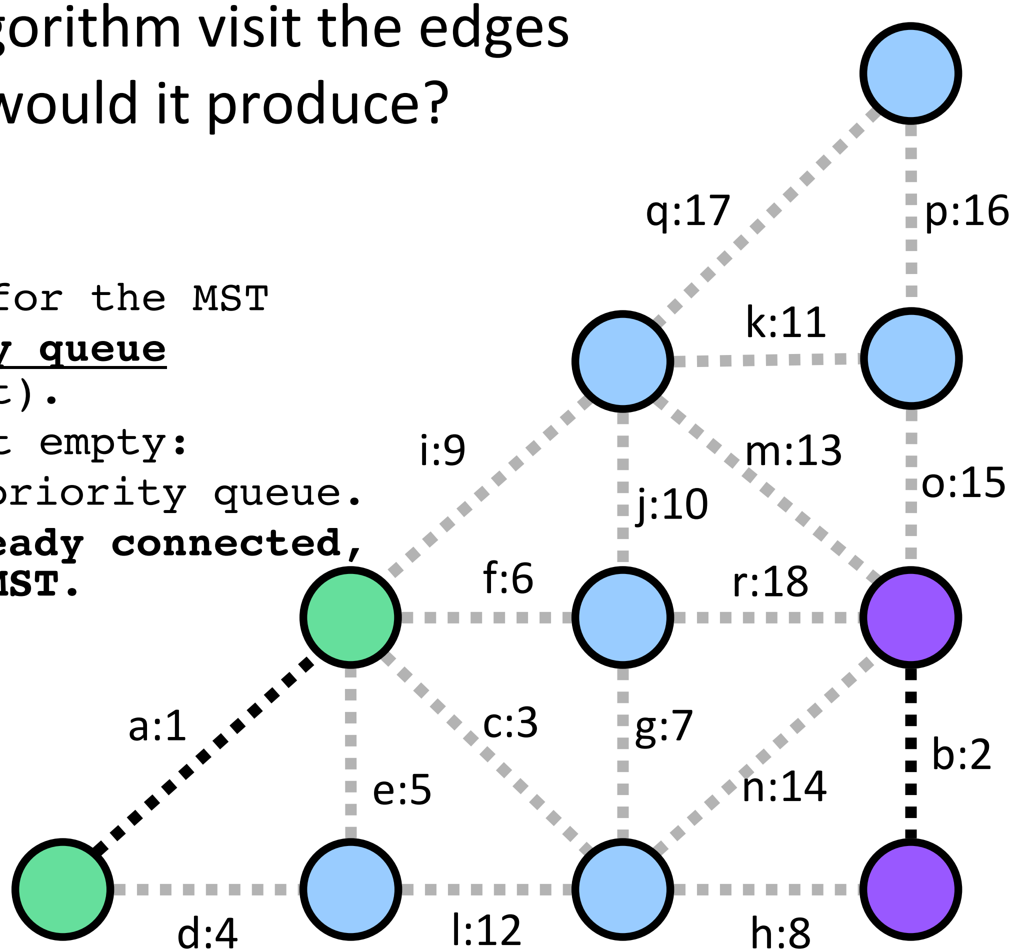


– pq = {c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
    Otherwise, skip the edge.
```

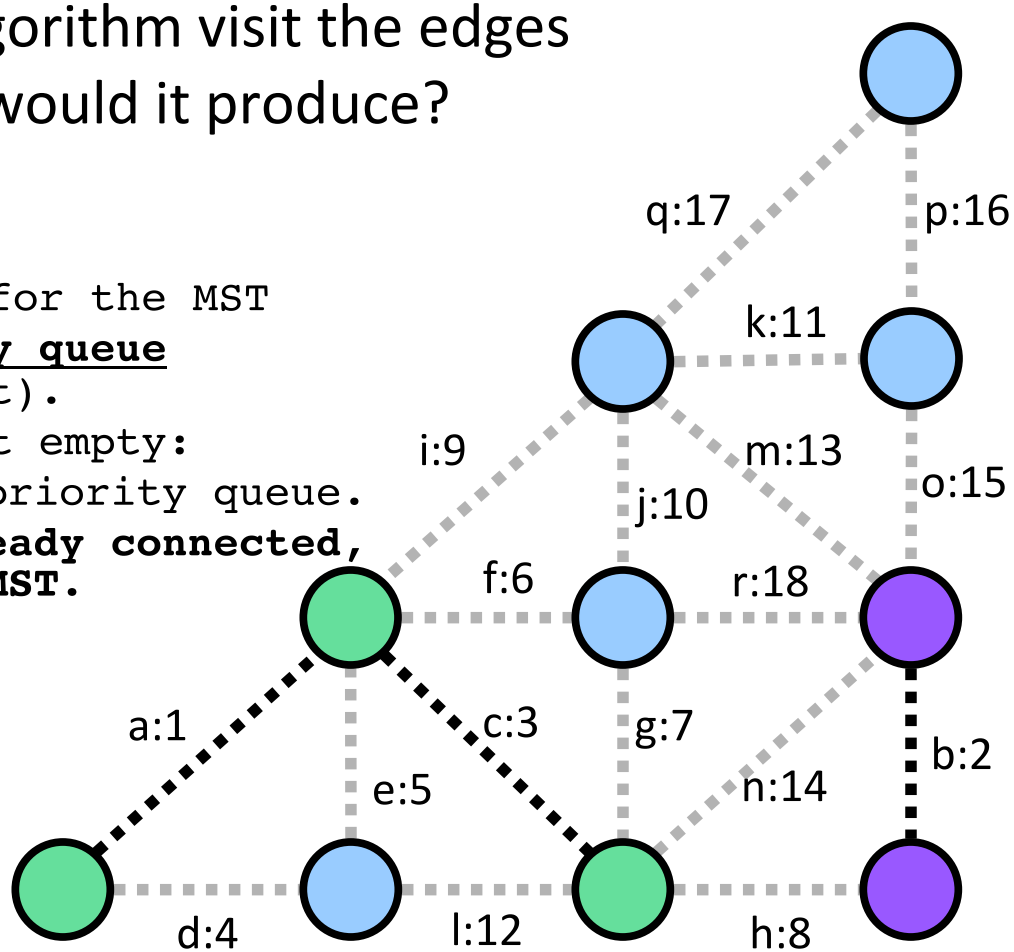


  - pq = {c:3, d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

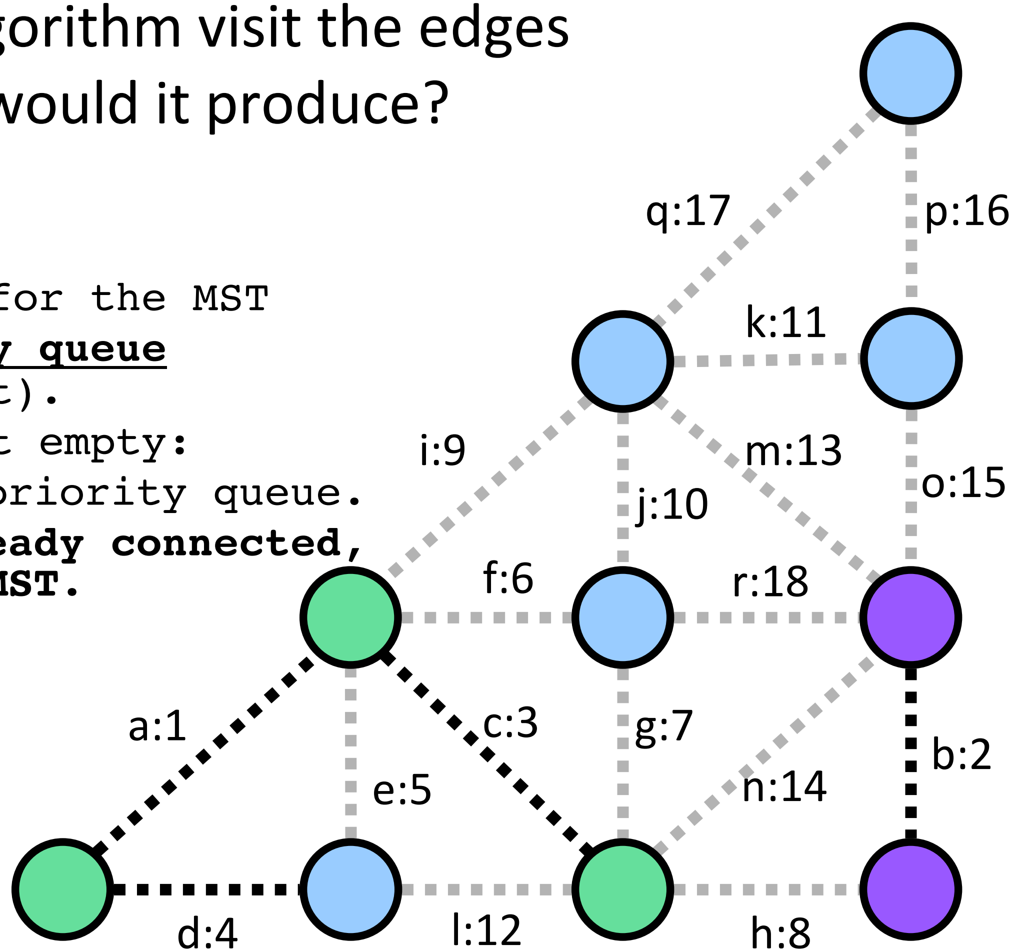


– pq = {d:4, e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

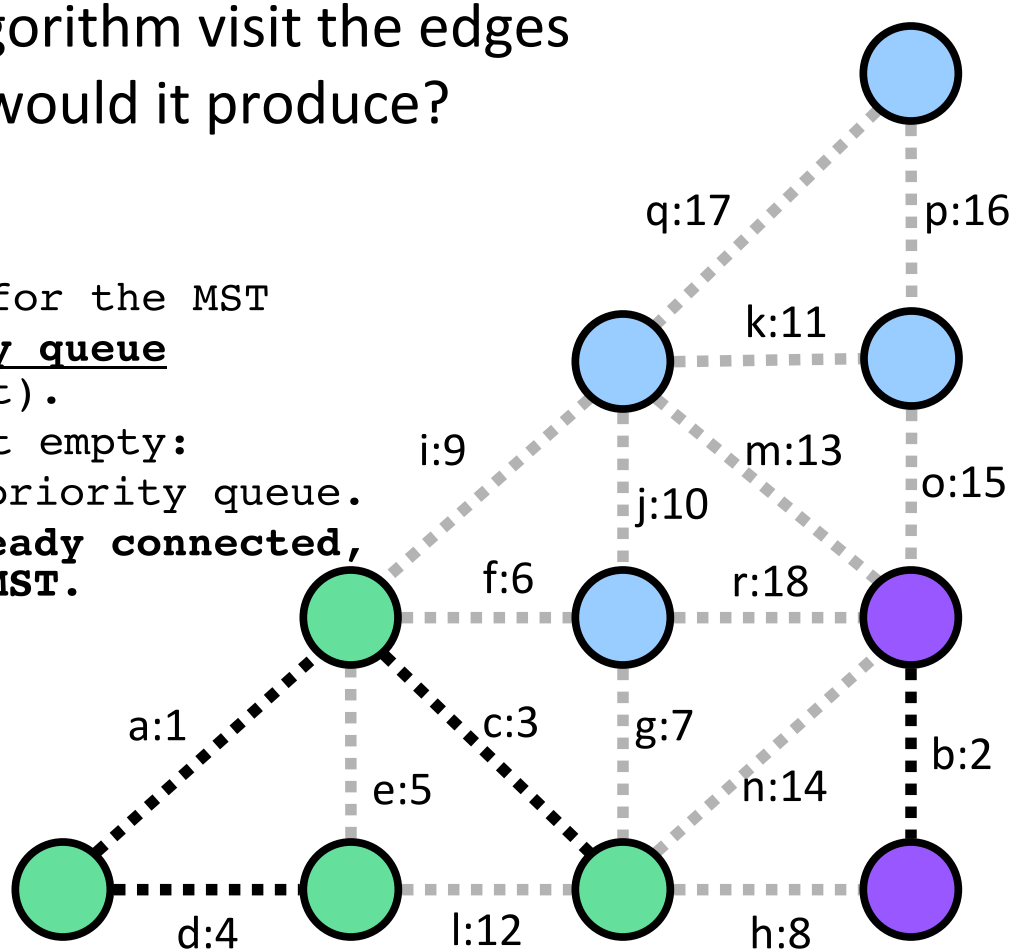


   – pq = {e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

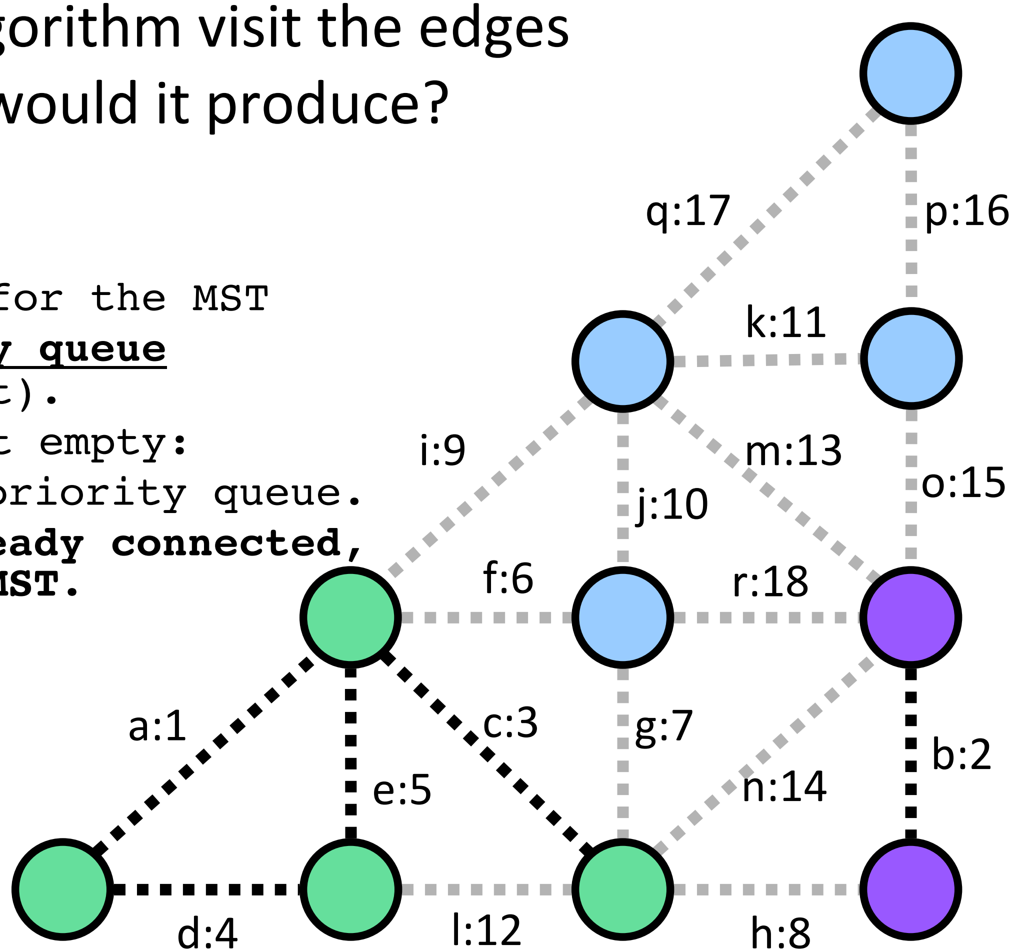


– pq = {e:5, f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

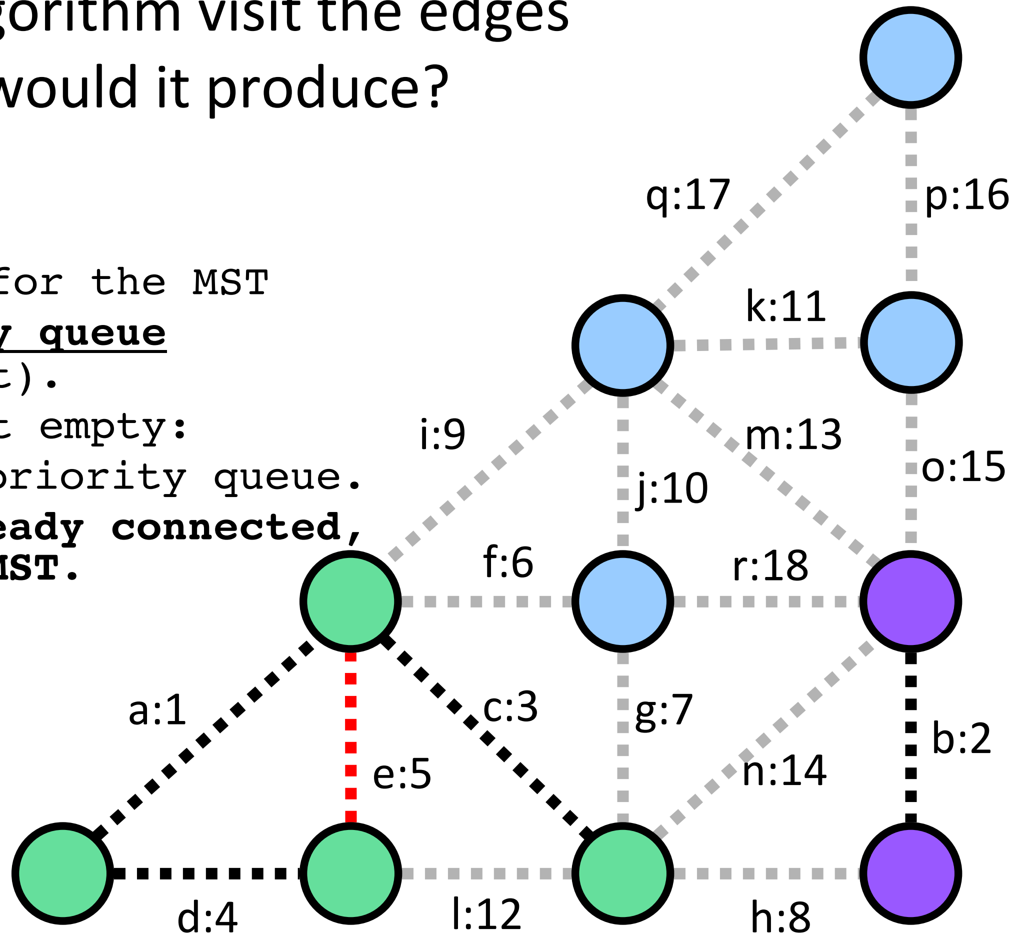


- pq = {f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

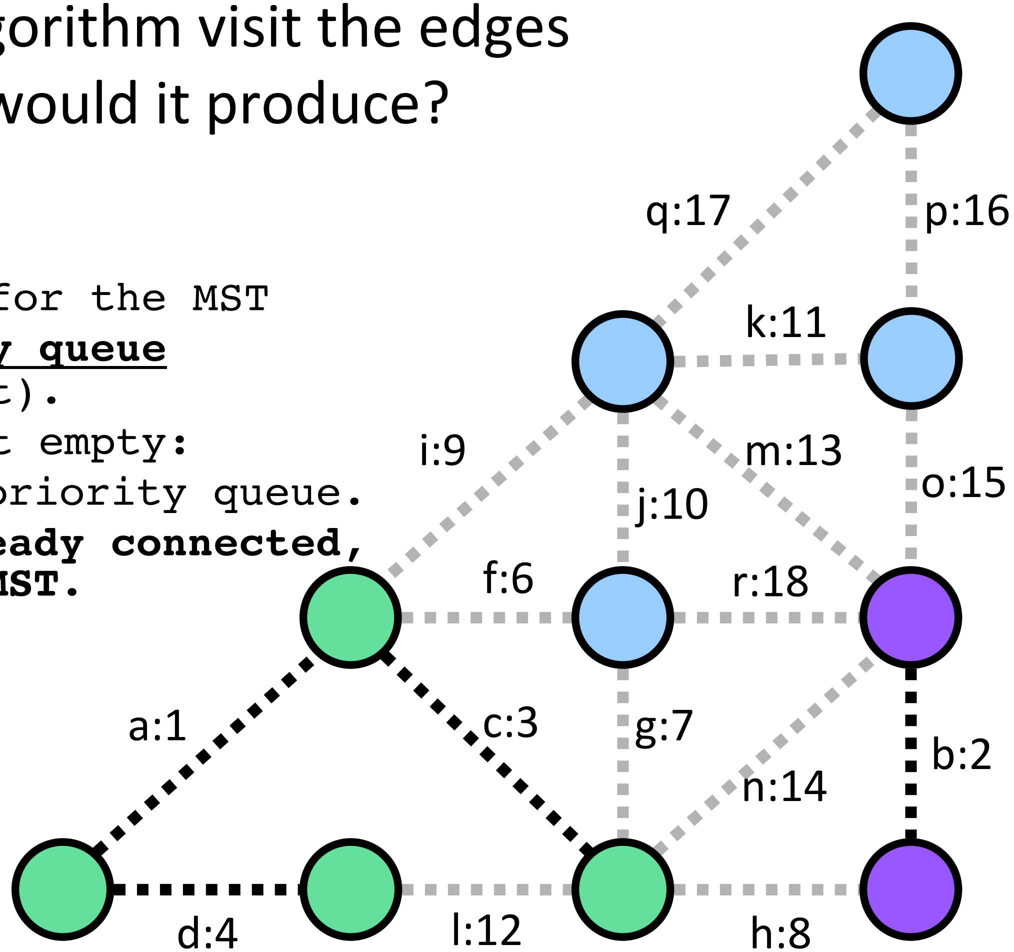


  - pq = {f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

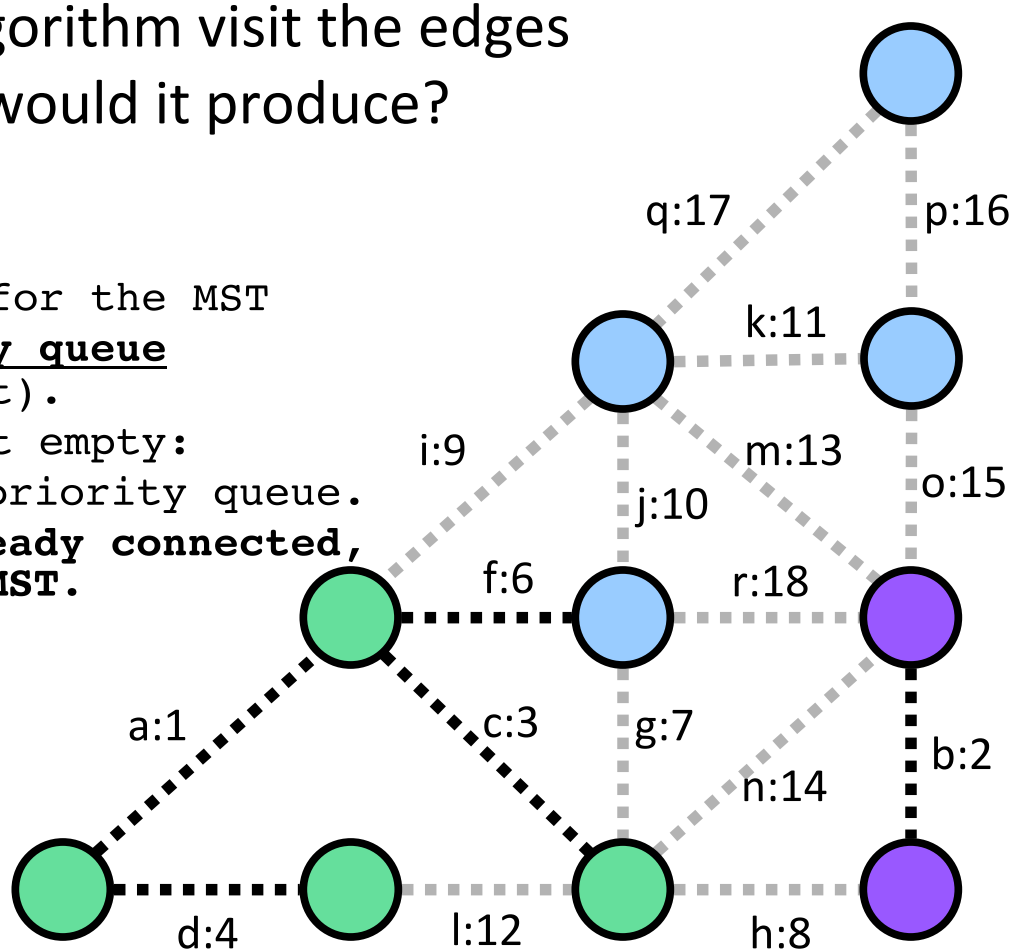


– pq = {f:6, g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

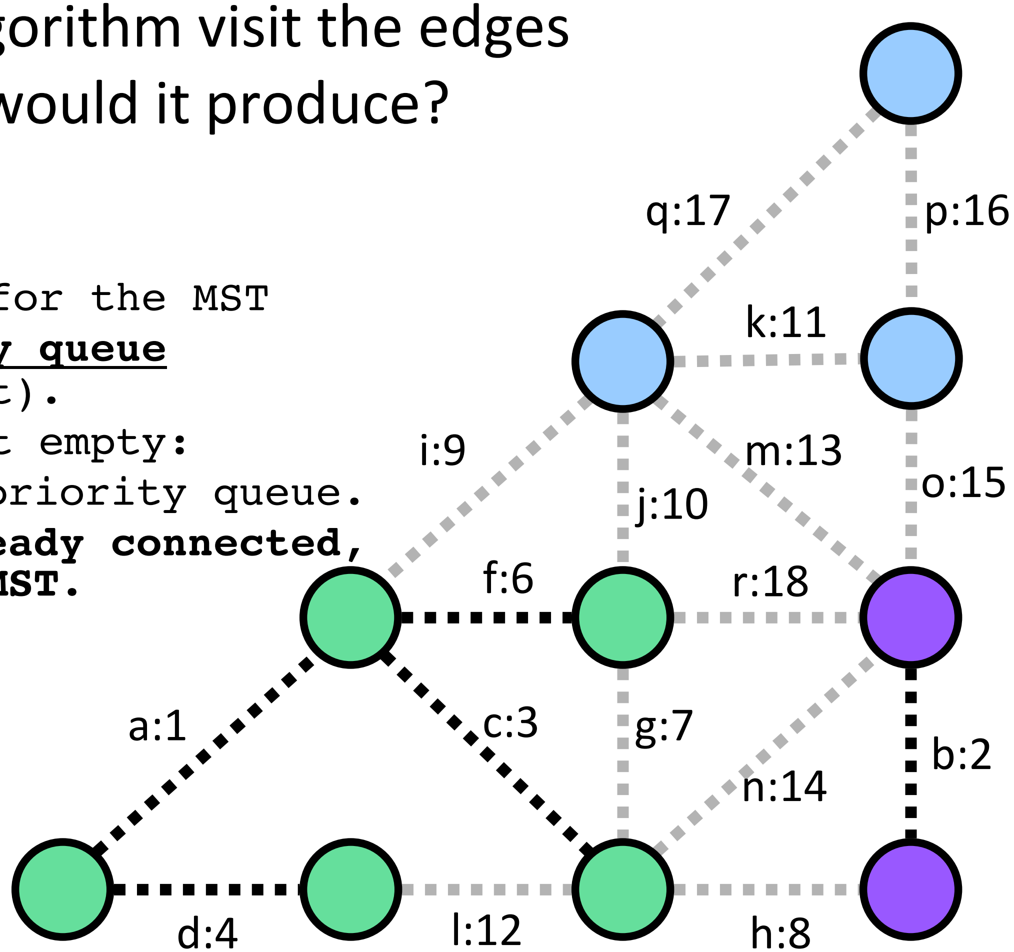


  - pq = {g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
    Otherwise, skip the edge.
```

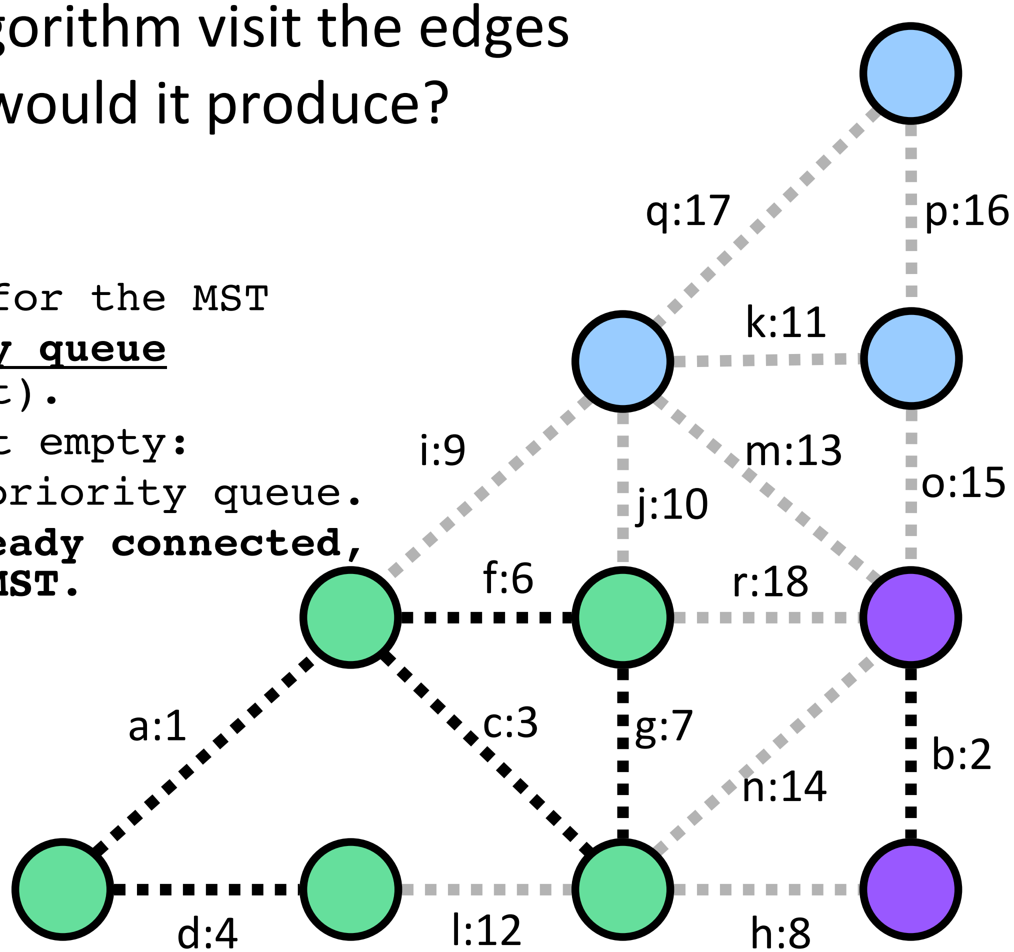


  - pq = {g:7, h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

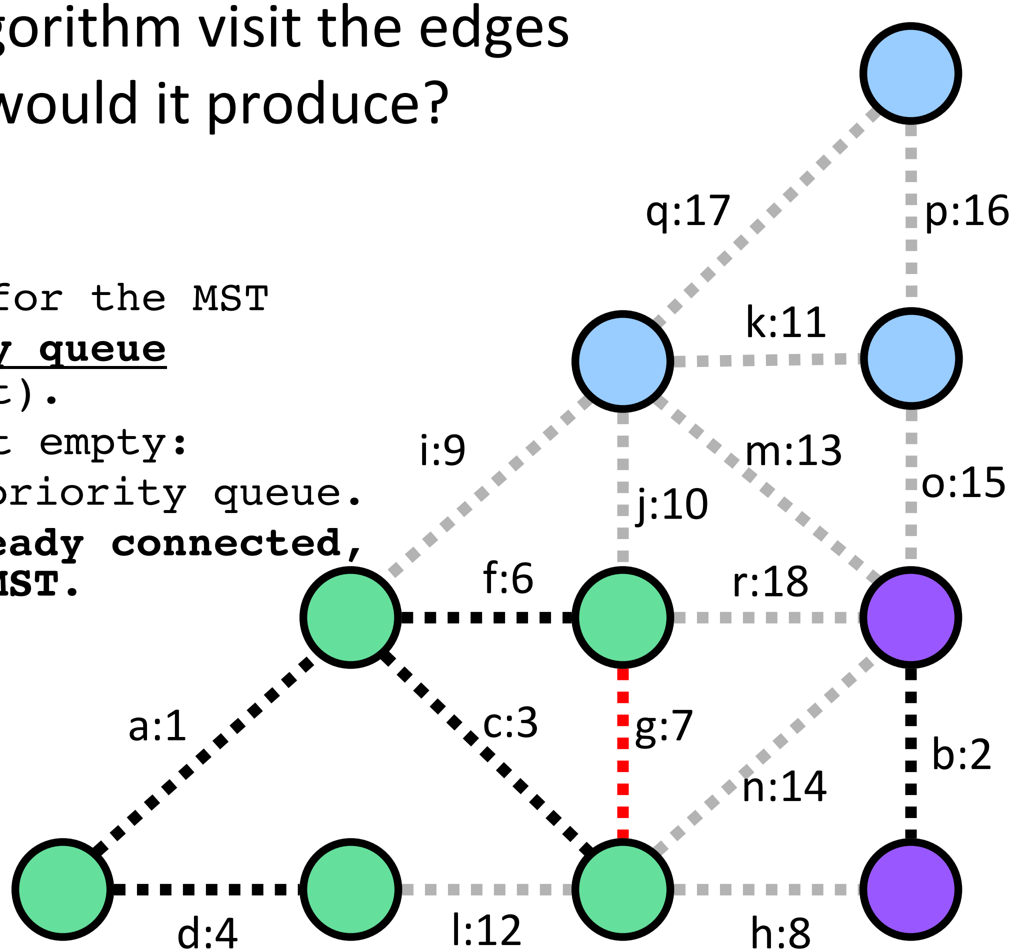


– pq = {h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

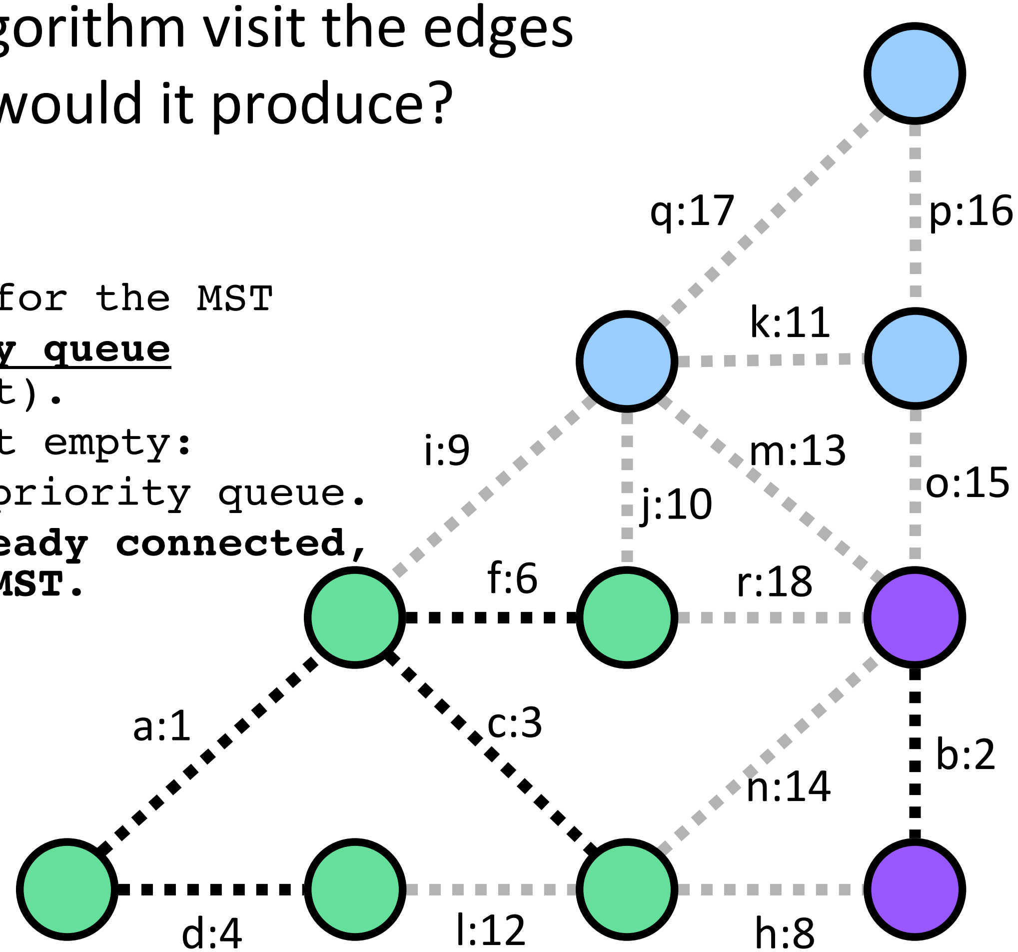


– pq = {h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

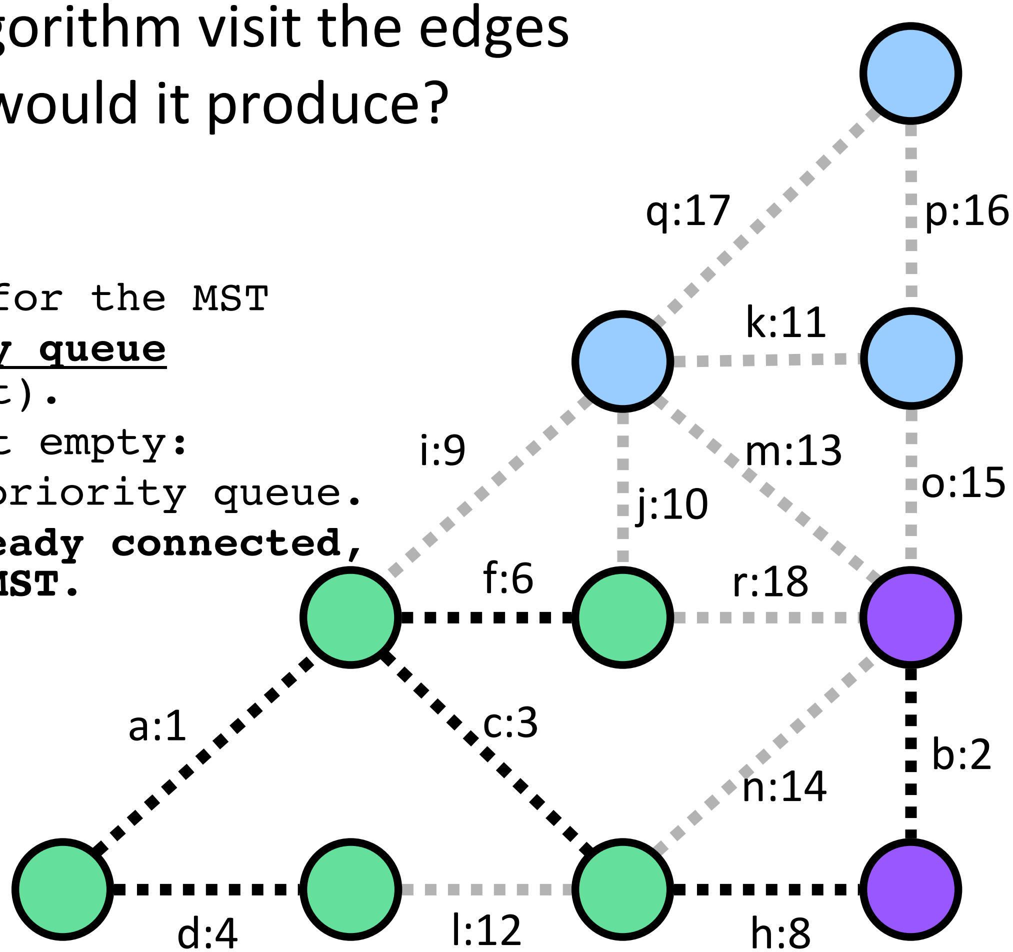


  - pq = {h:8, i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
    Otherwise, skip the edge.
```

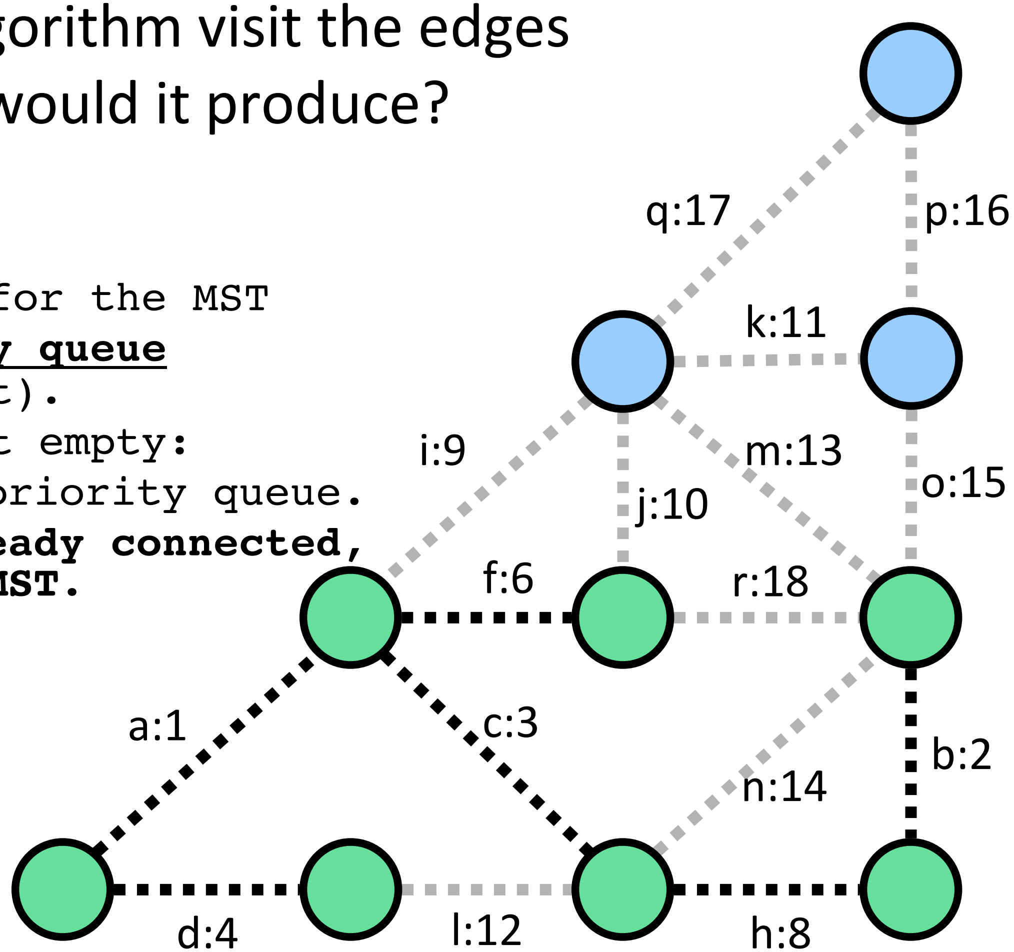


  - pq = {i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

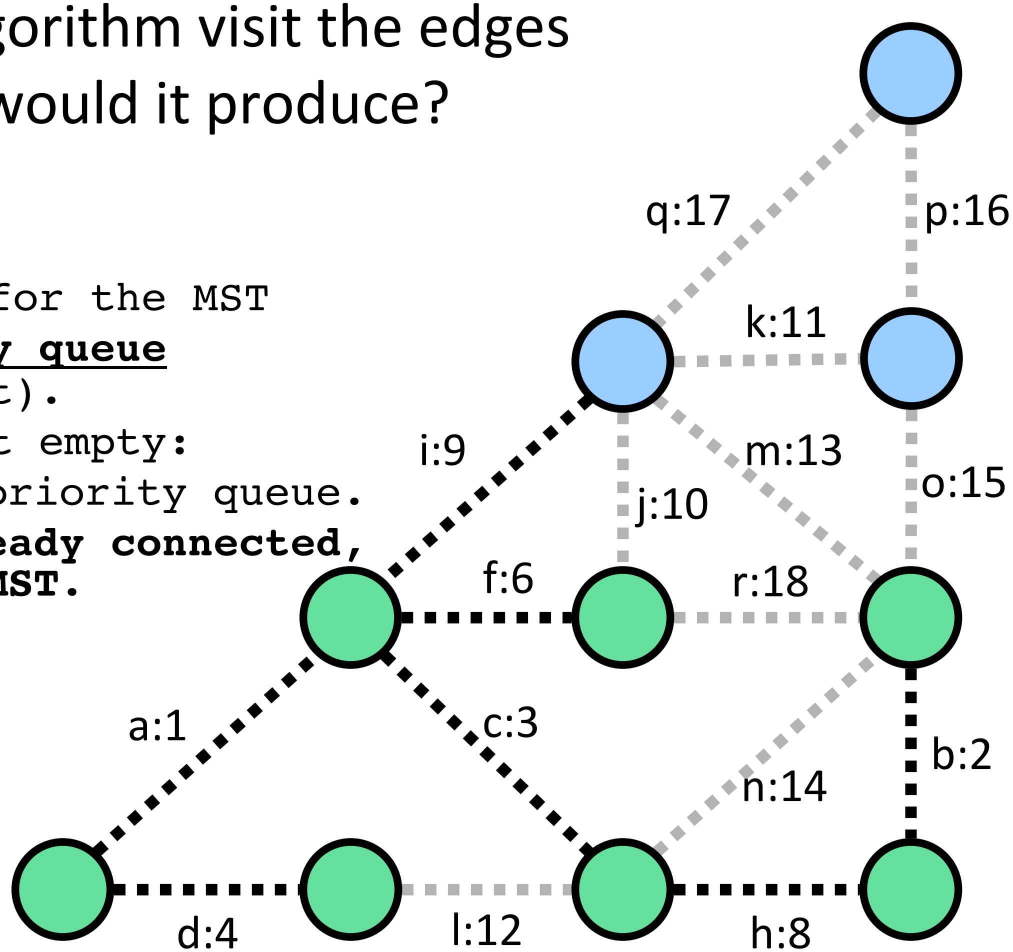


  - pq = {i:9, j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

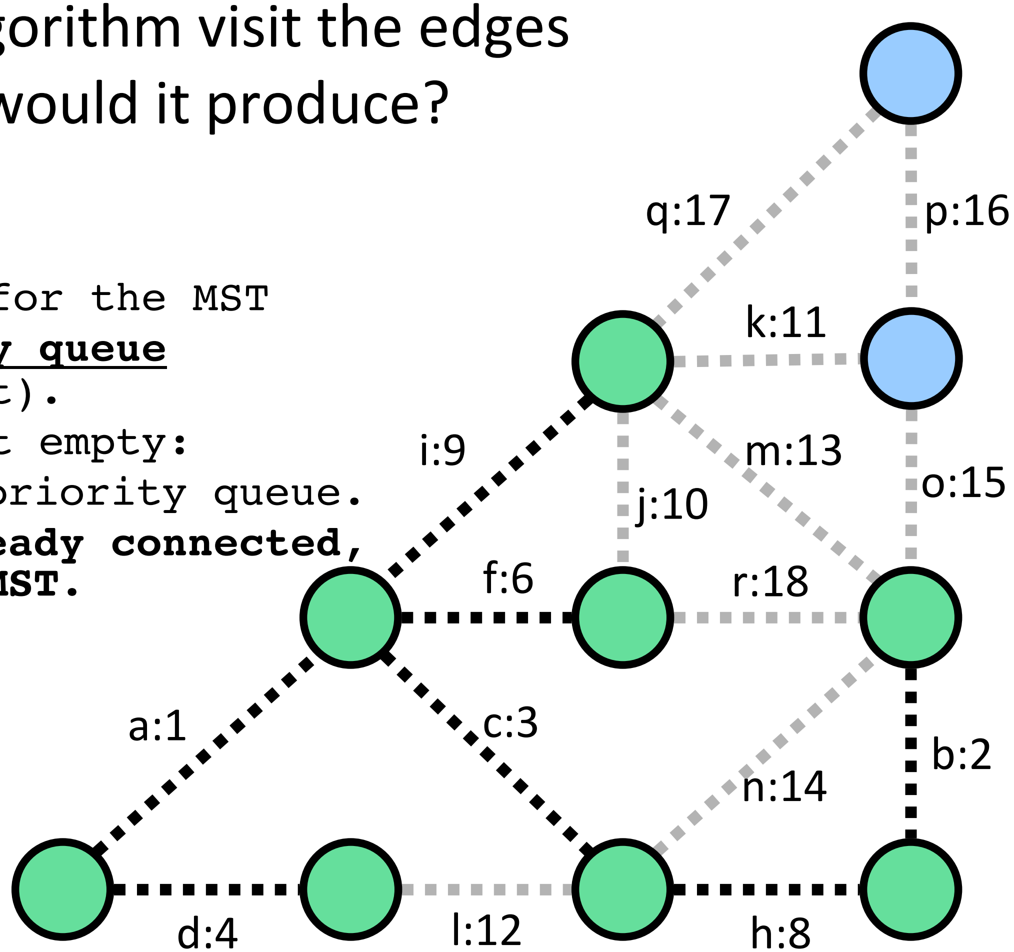


  - pq = {j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

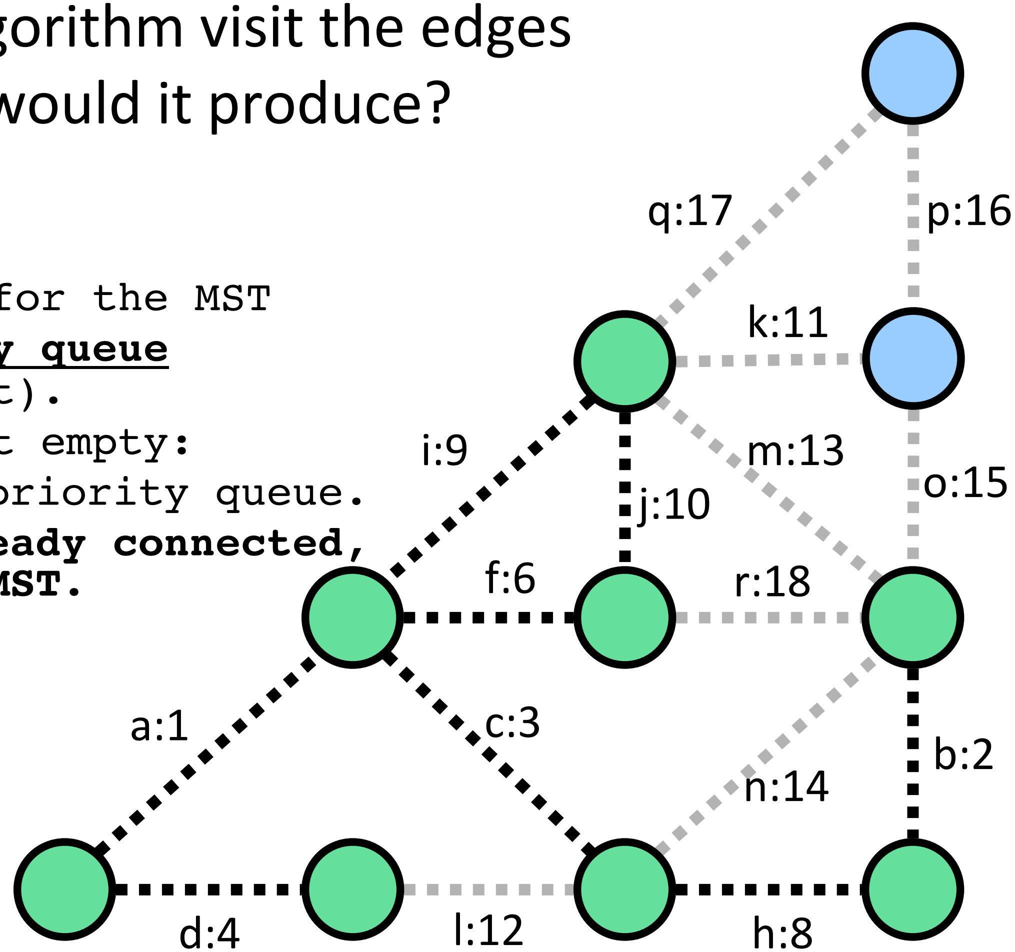


  - pq = {j:10, k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

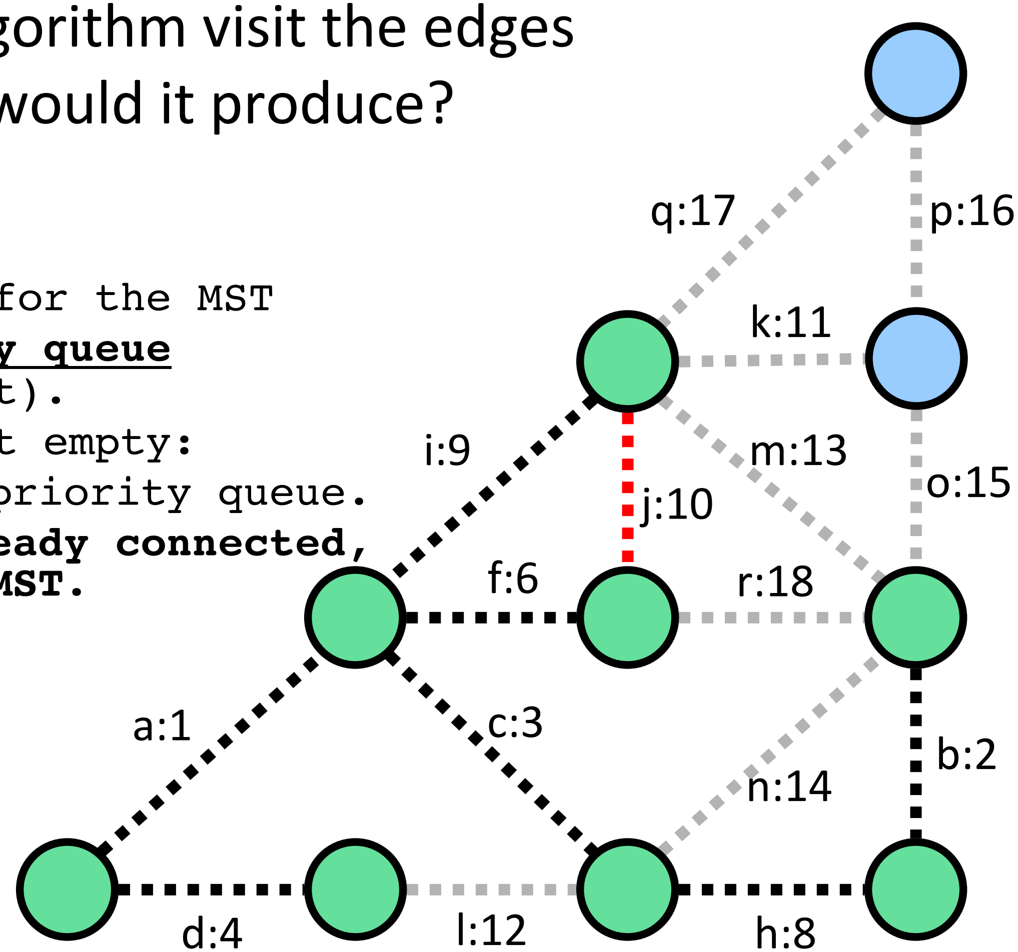


– pq = {k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

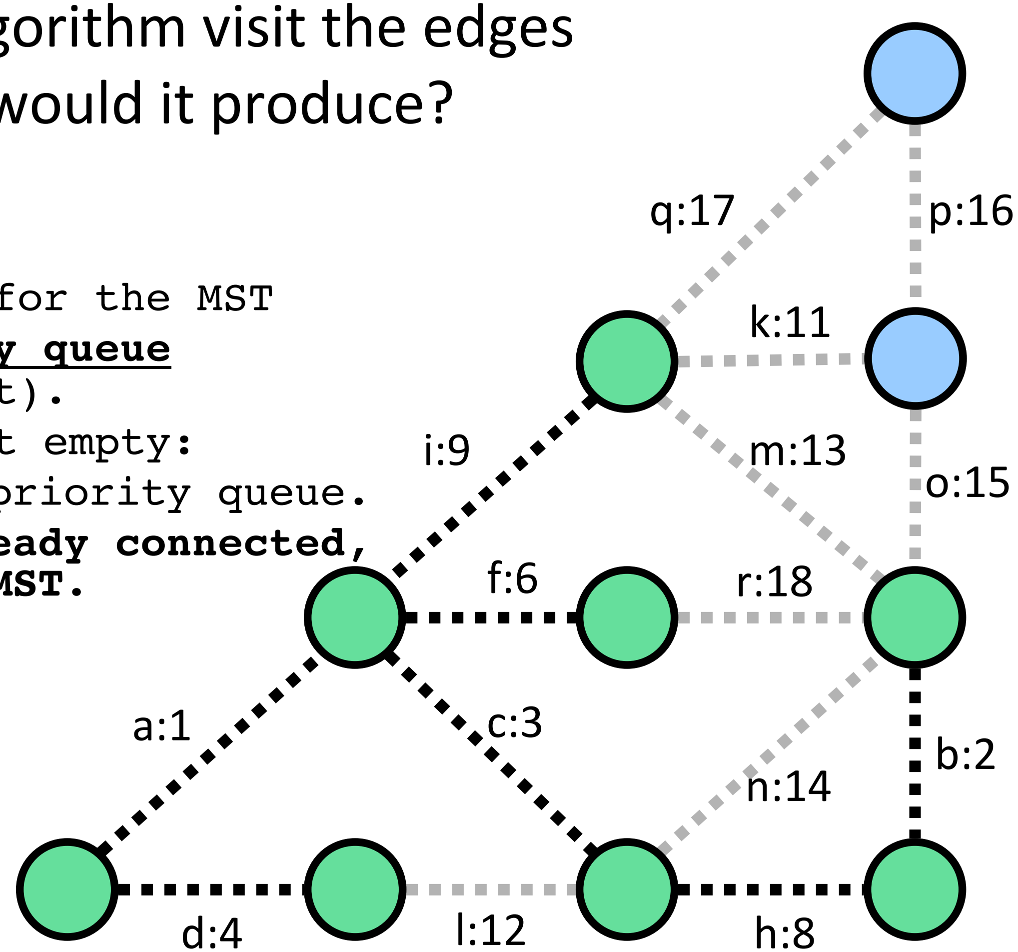


  - pq = {k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

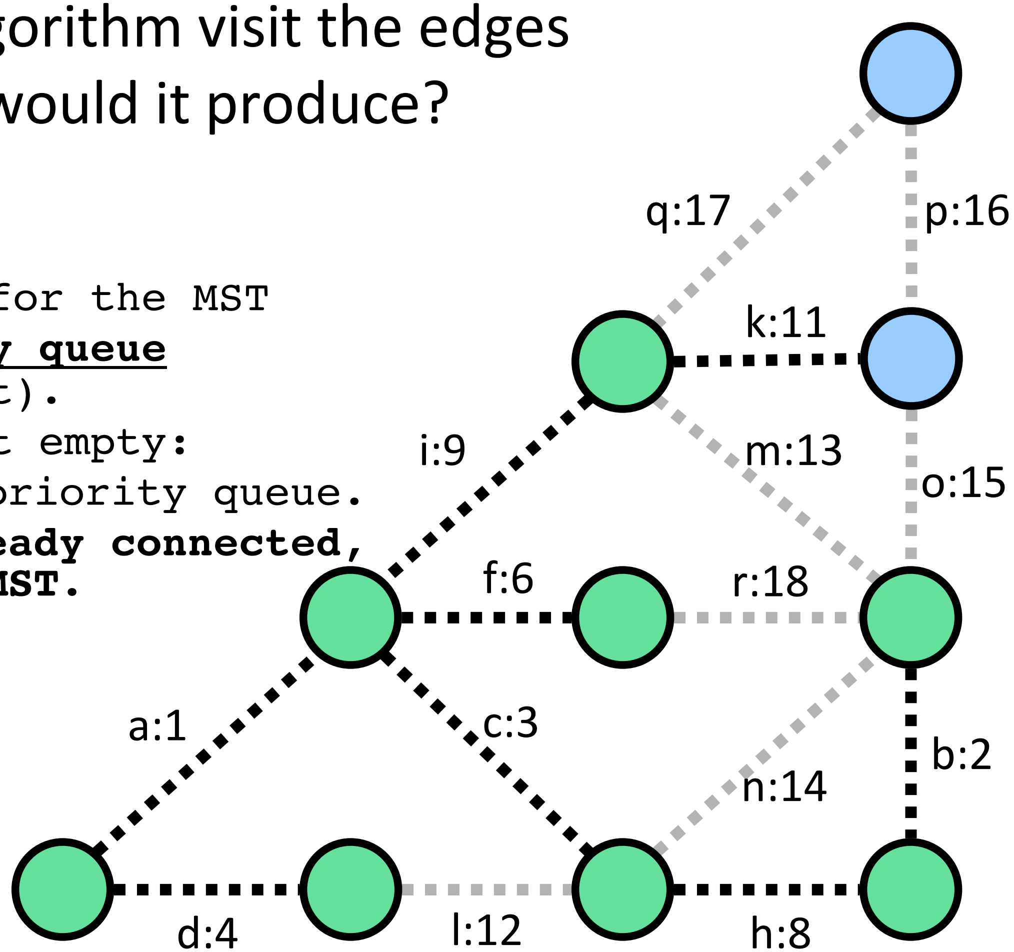


  - pq = {k:11, l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

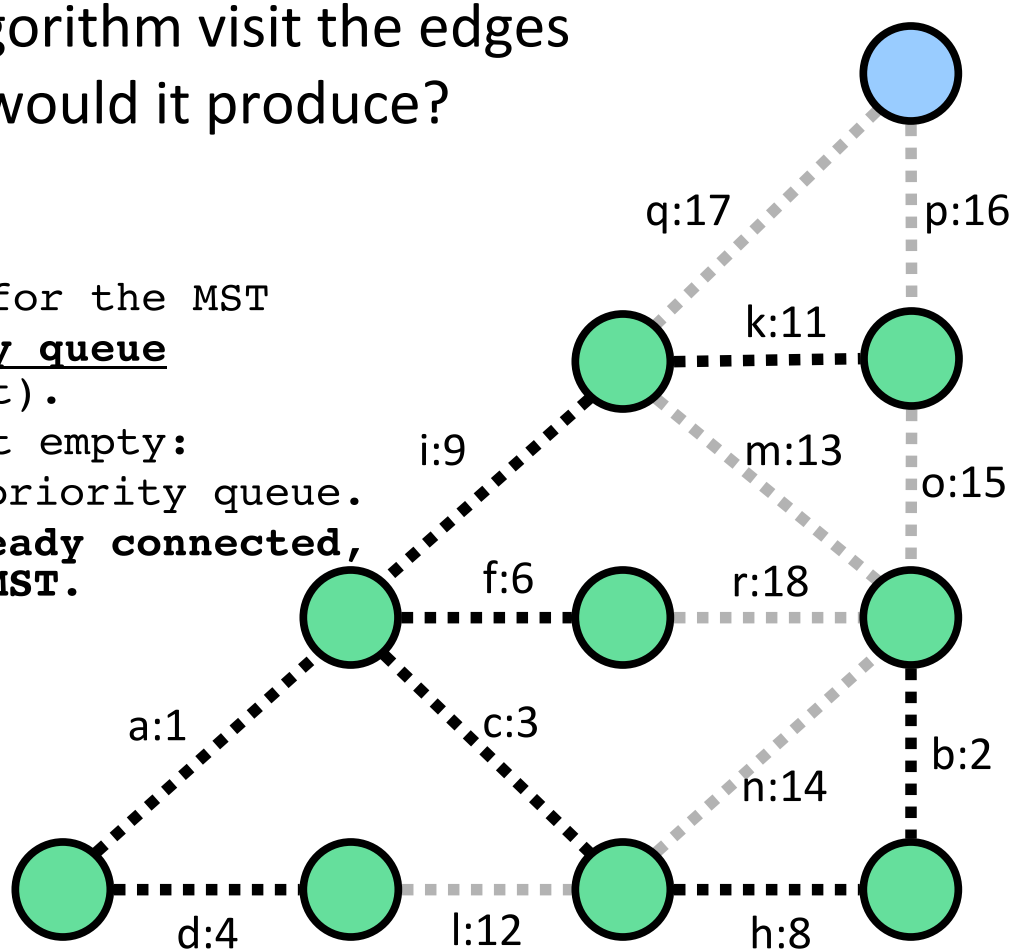


  – pq = {l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

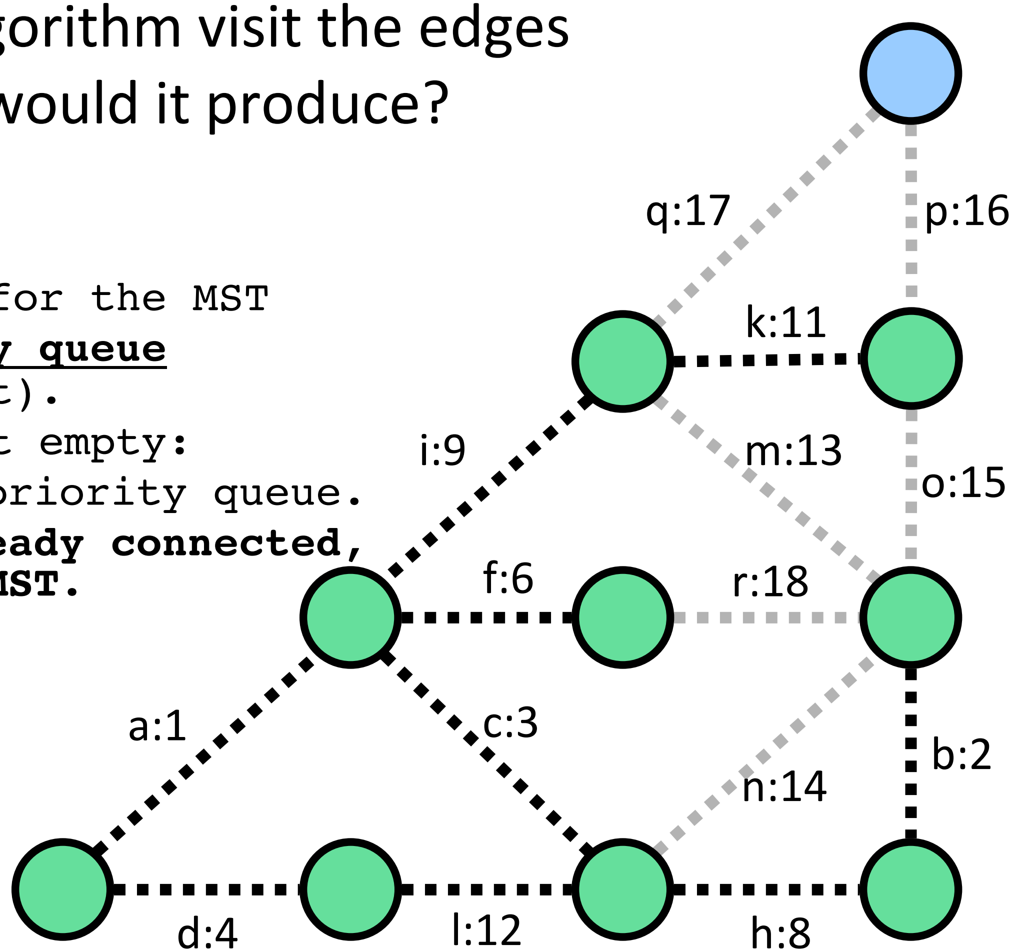


- pq = {l:12, m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

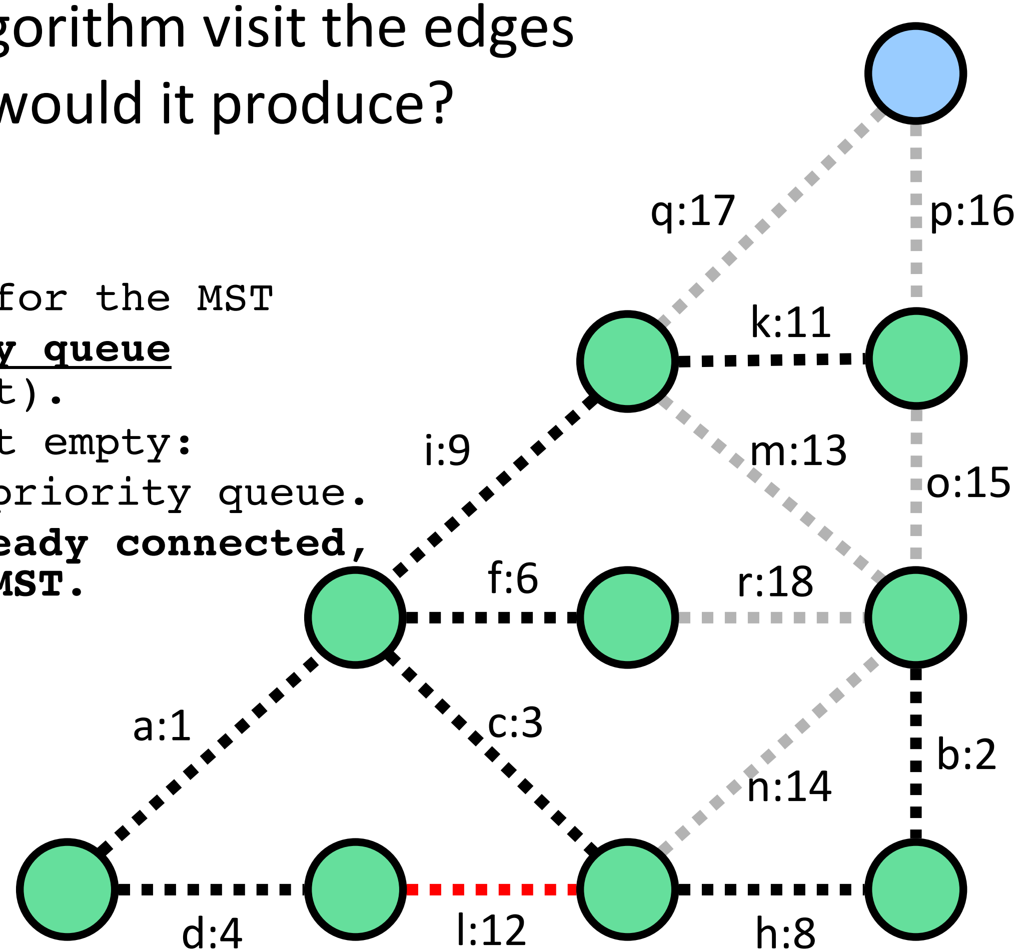


  - pq = {m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

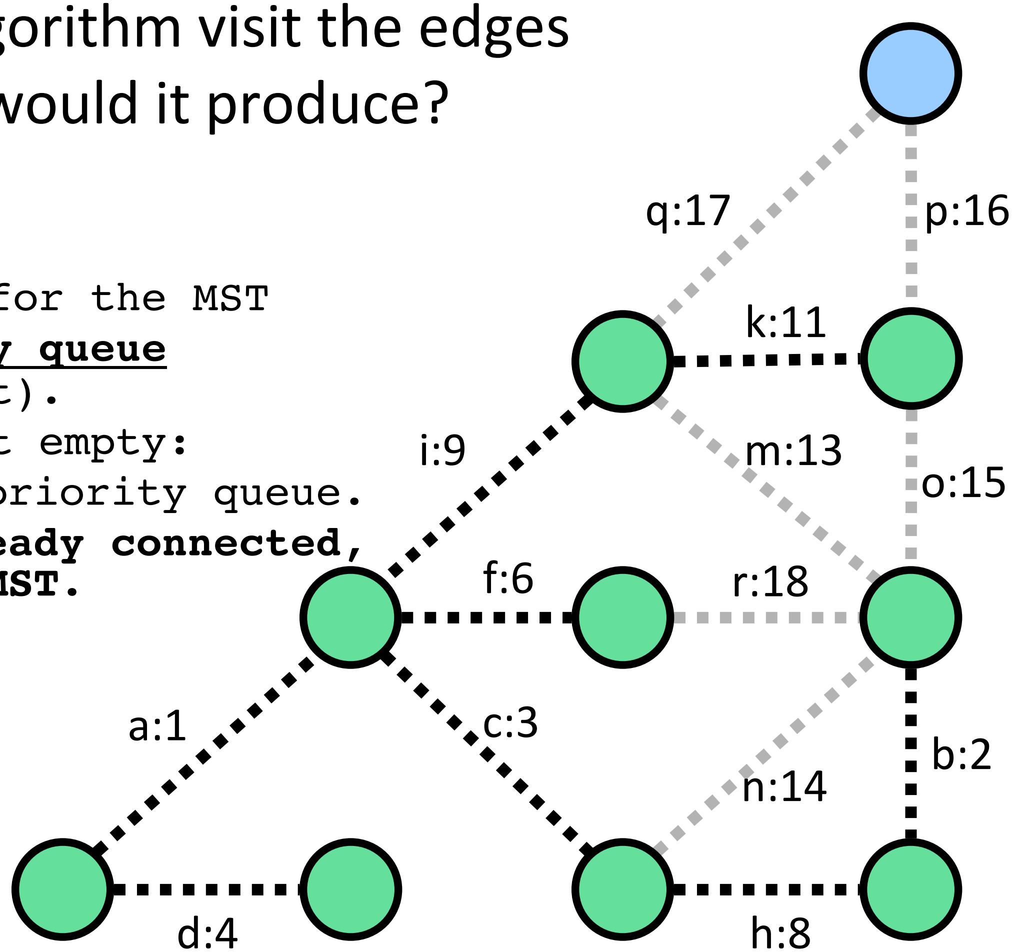


  - pq = {m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

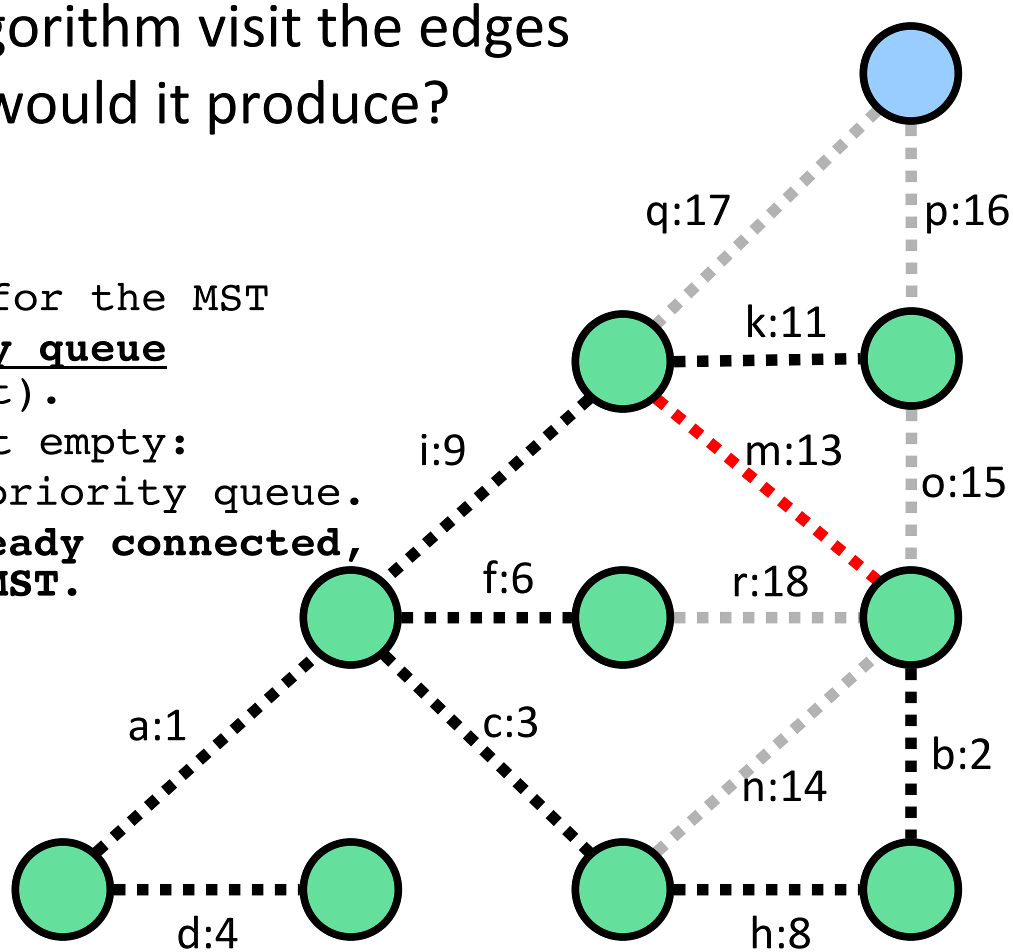


  – pq = {m:13, n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

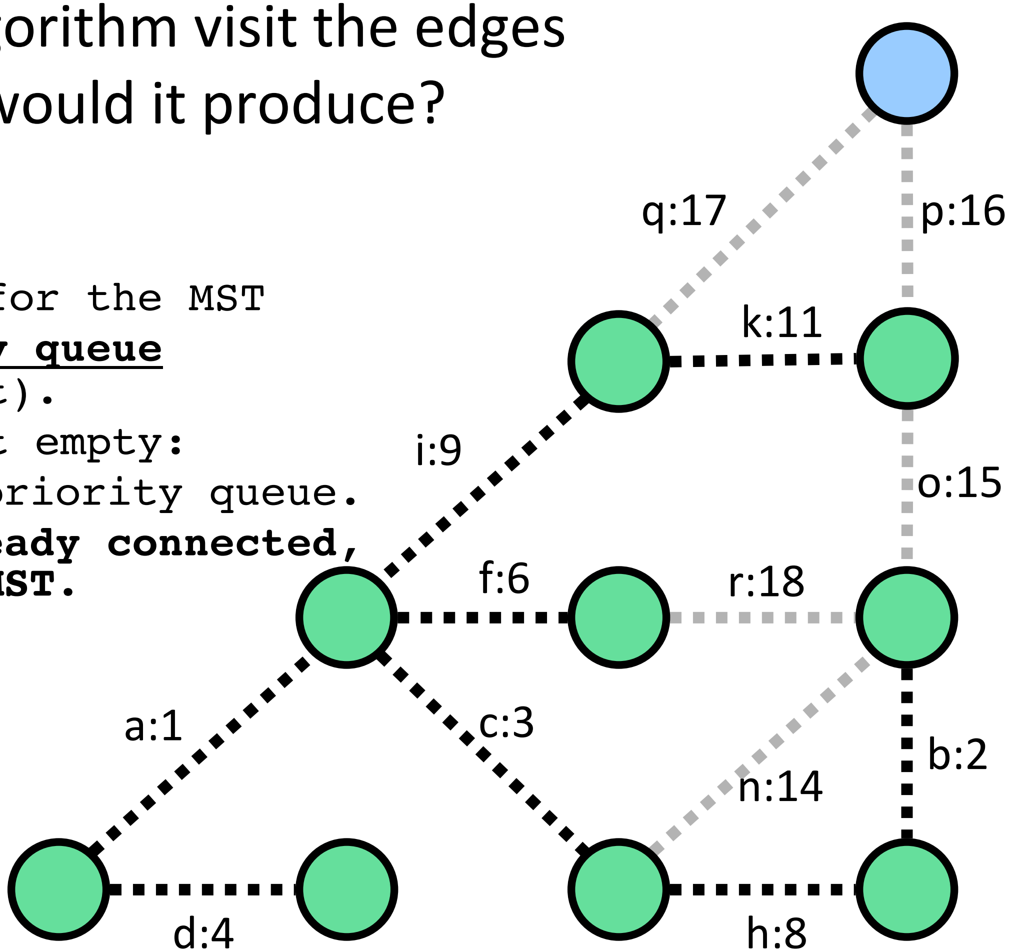


  – pq = {n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

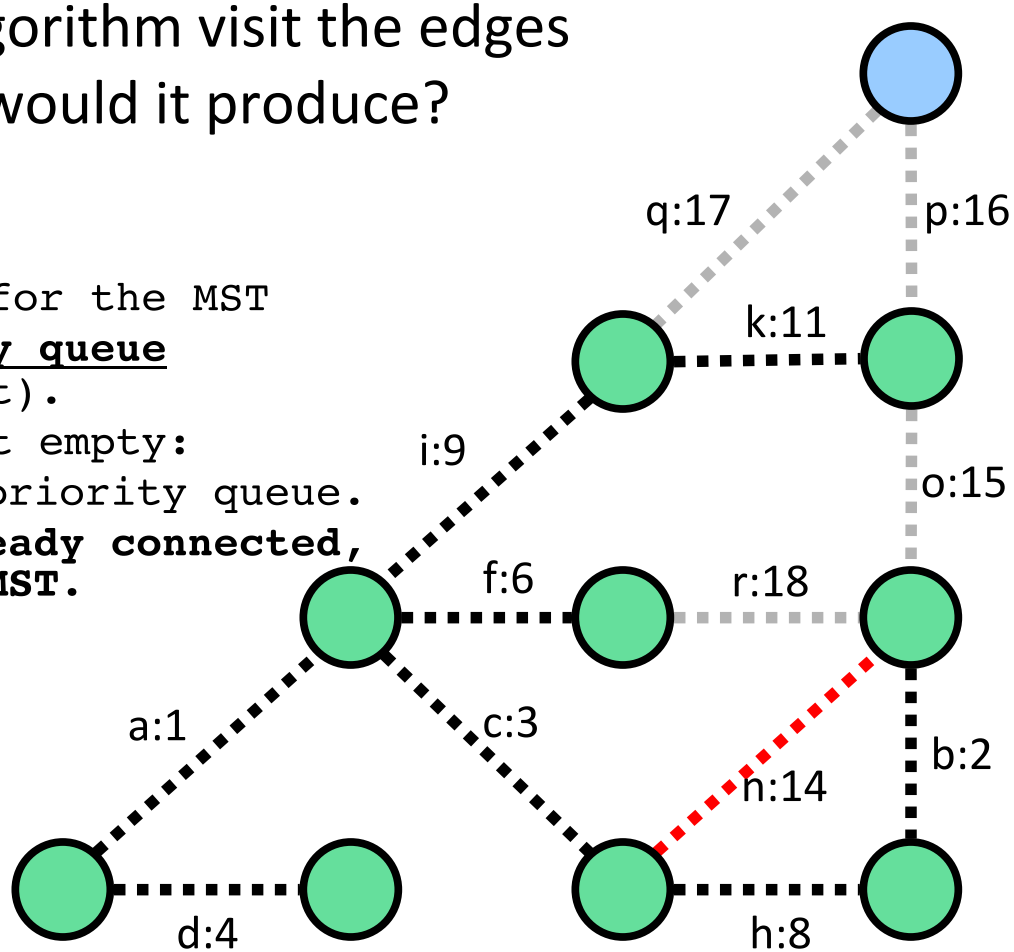


  – pq = {n:14, o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
    Otherwise, skip the edge.
```

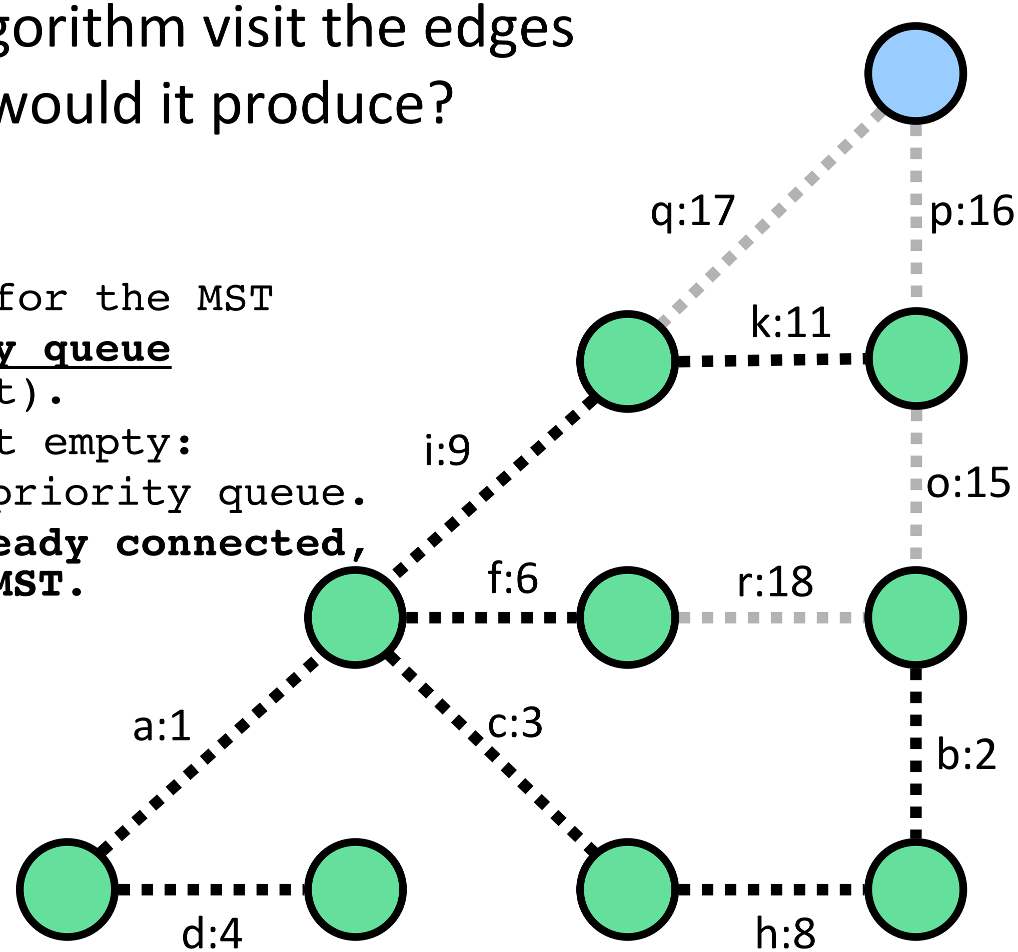


  - pq = {o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

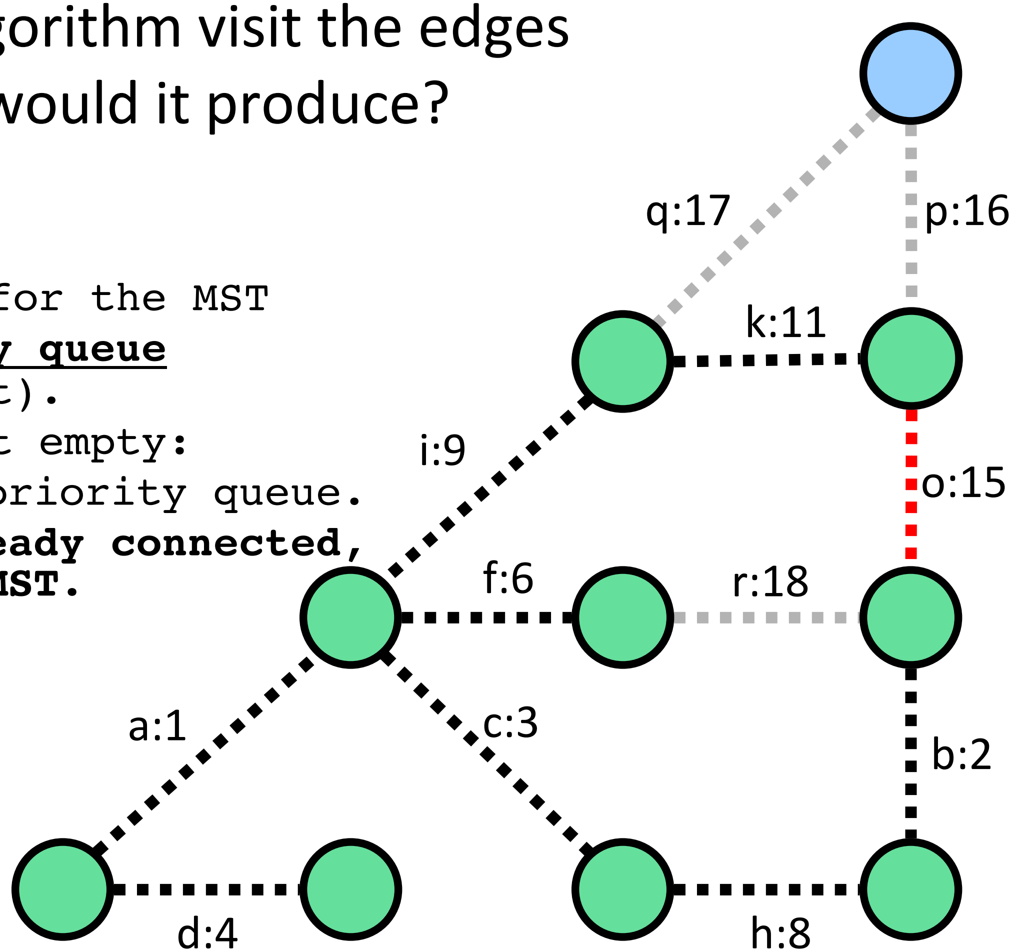


  - pq = {o:15, p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

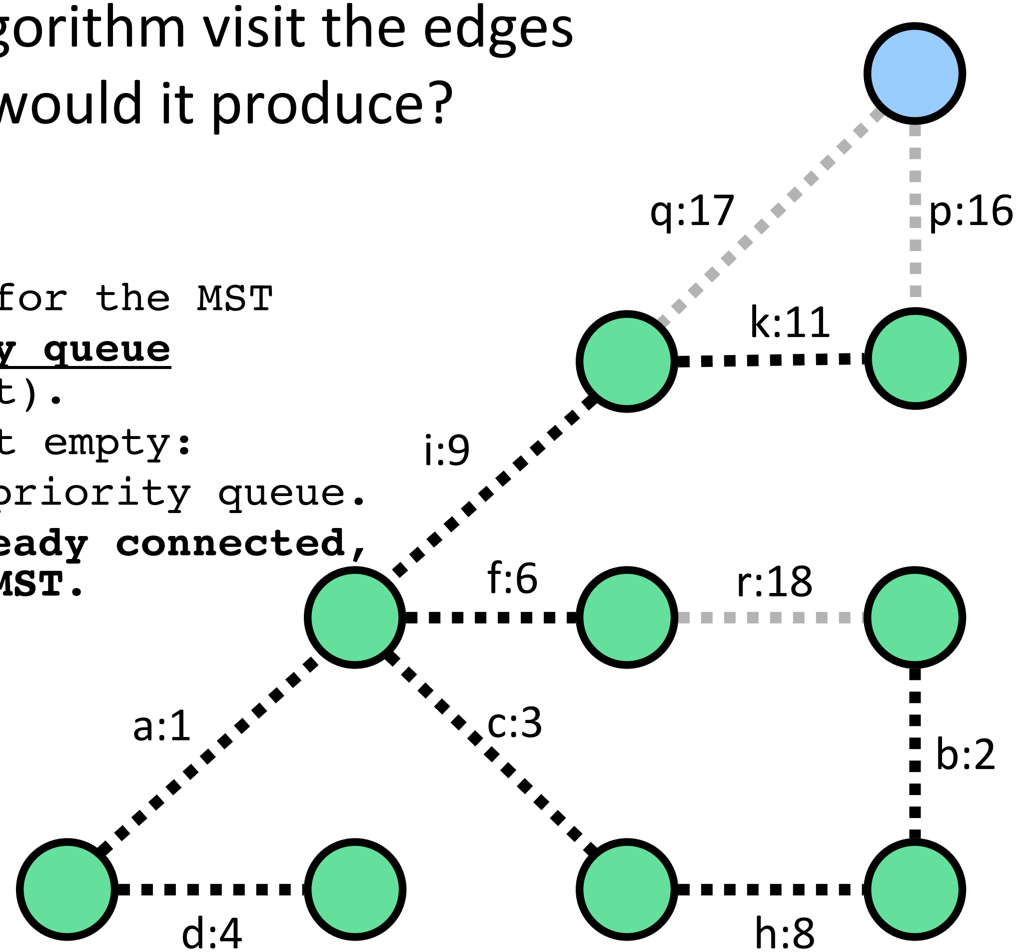


  - pq = {p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

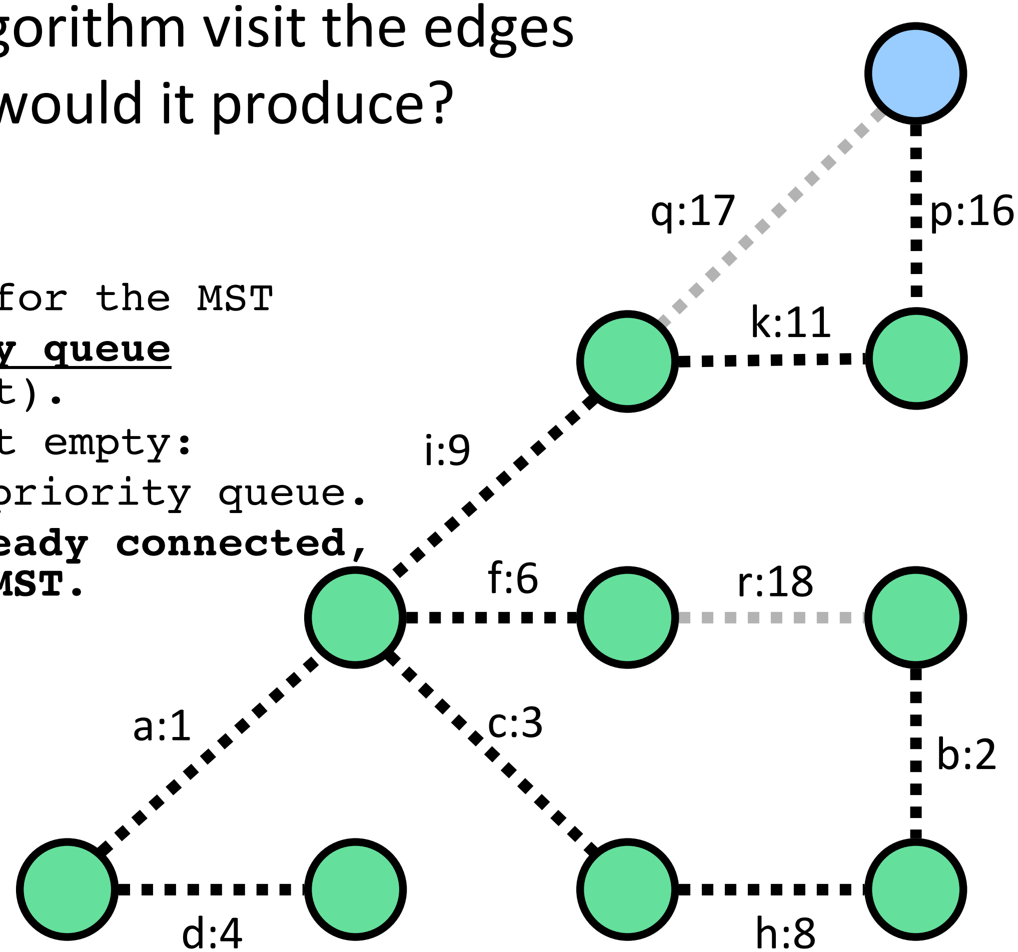


  - pq = {p:16, q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
    Otherwise, skip the edge.
```

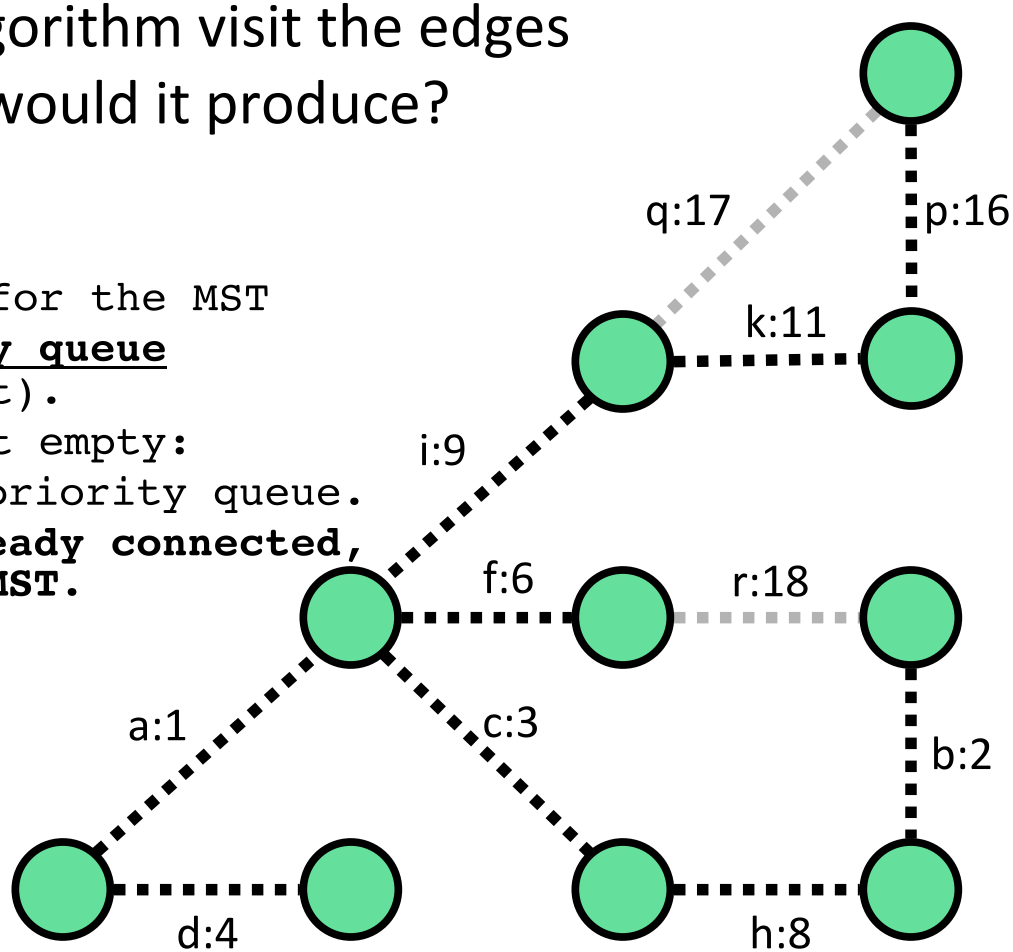


– pq = {q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

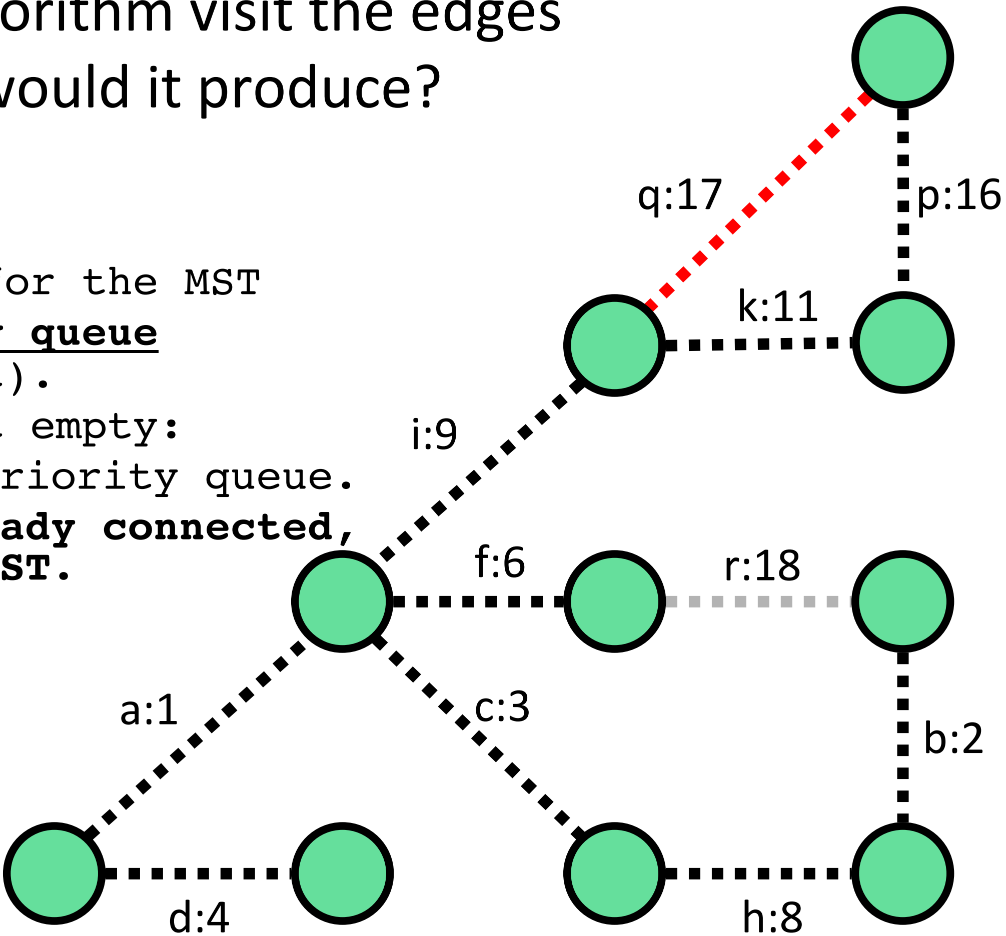


  - pq = {q:17, r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

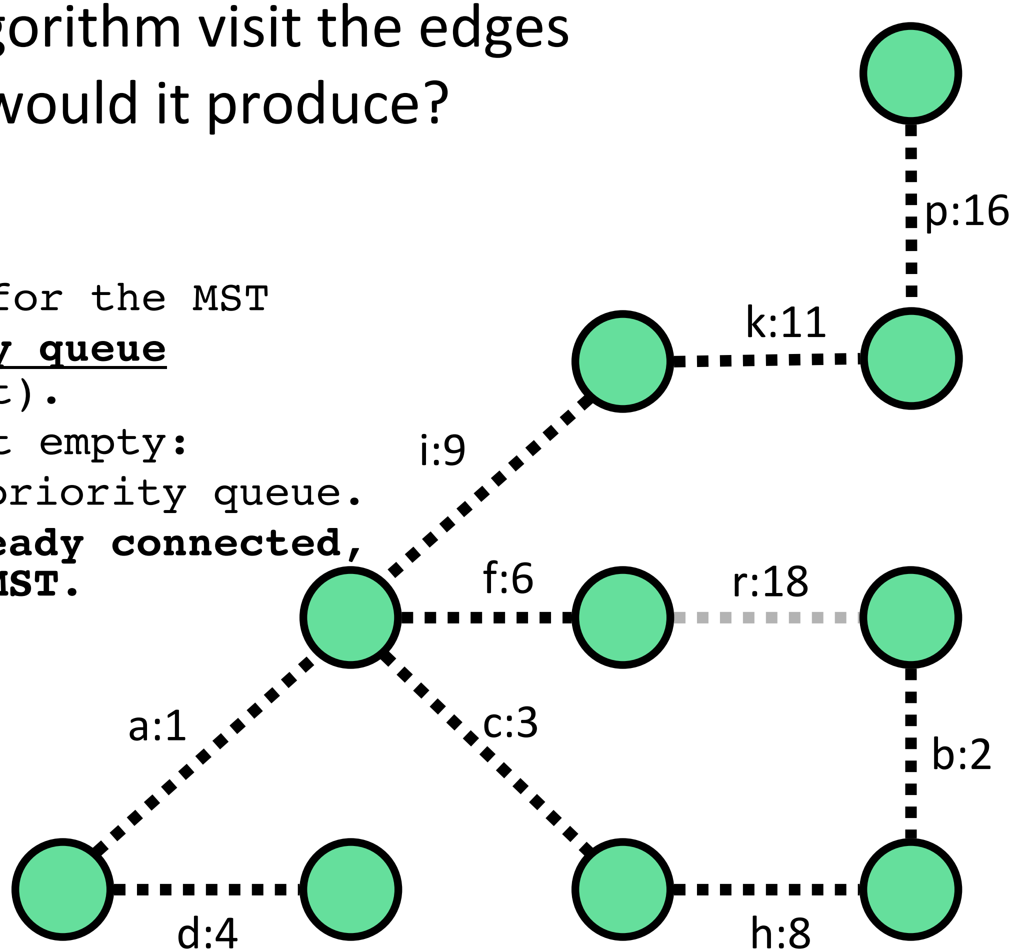


  - pq = {r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

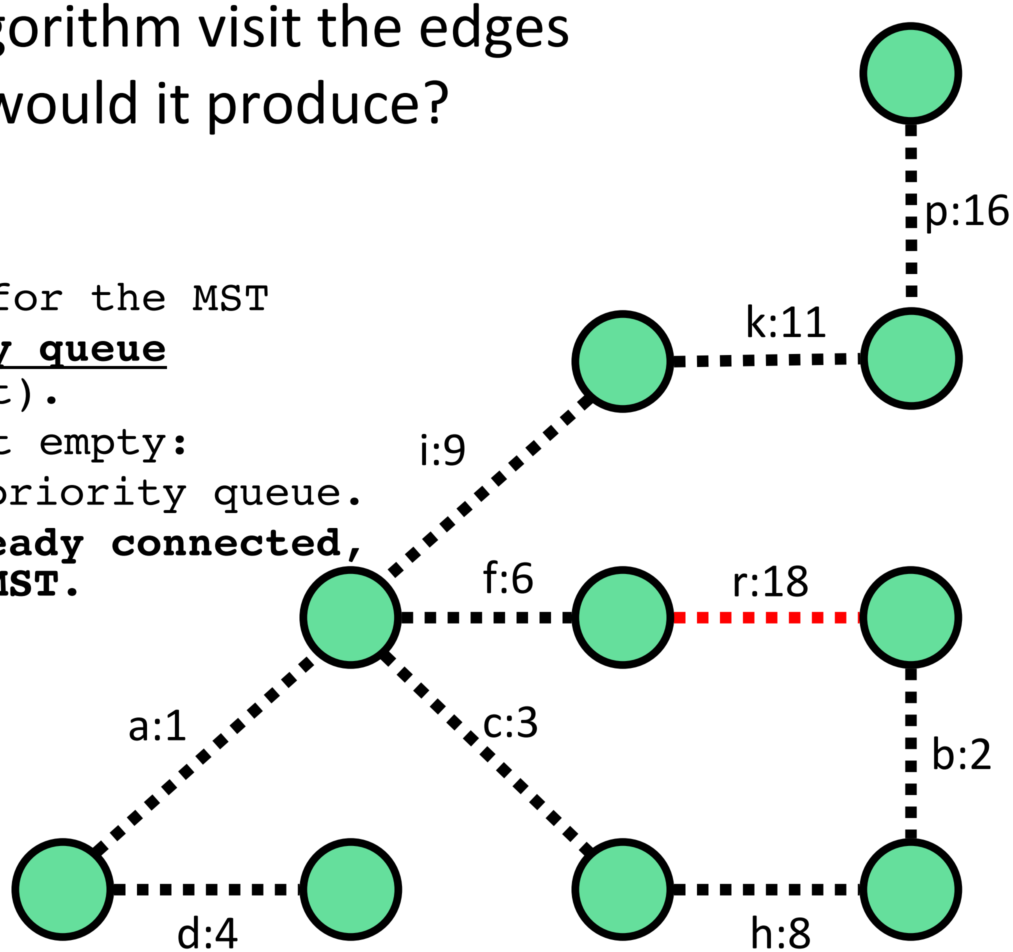


  – pq = {r:18}

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

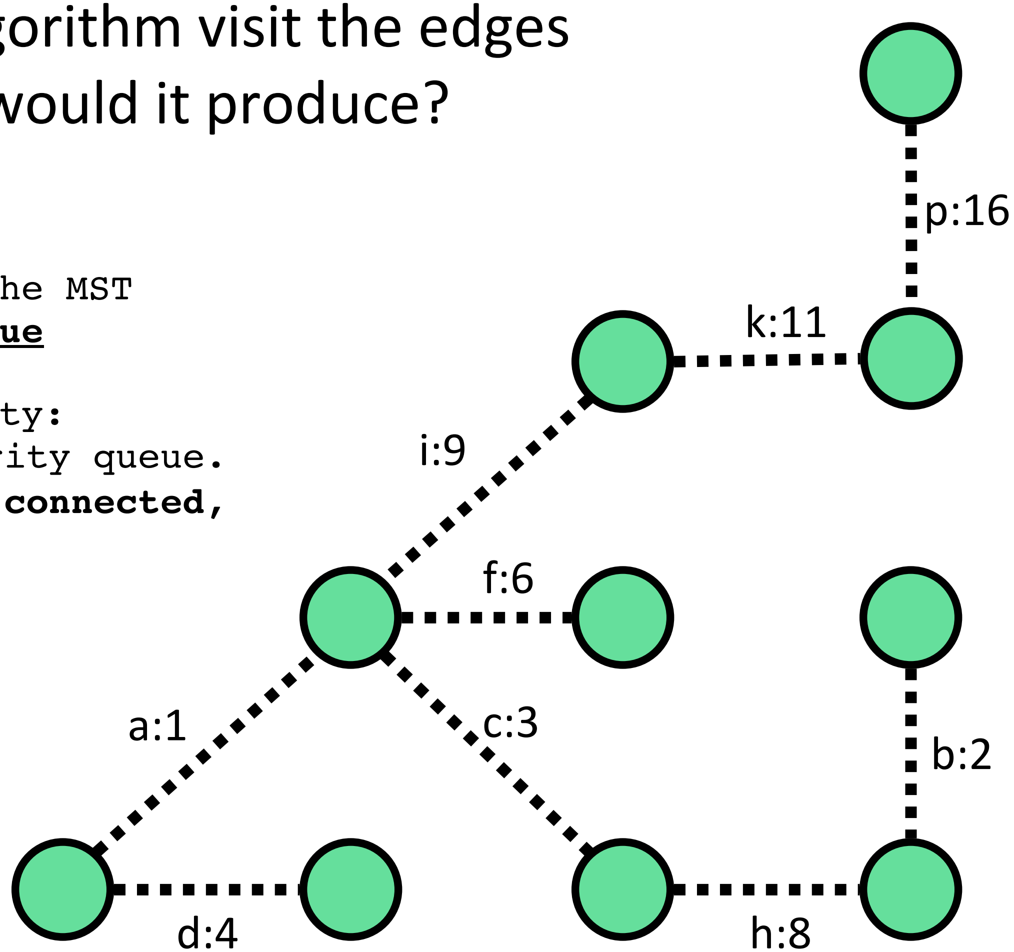


  - pq = { }

Kruskal example

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

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
     Otherwise, skip the edge.
```

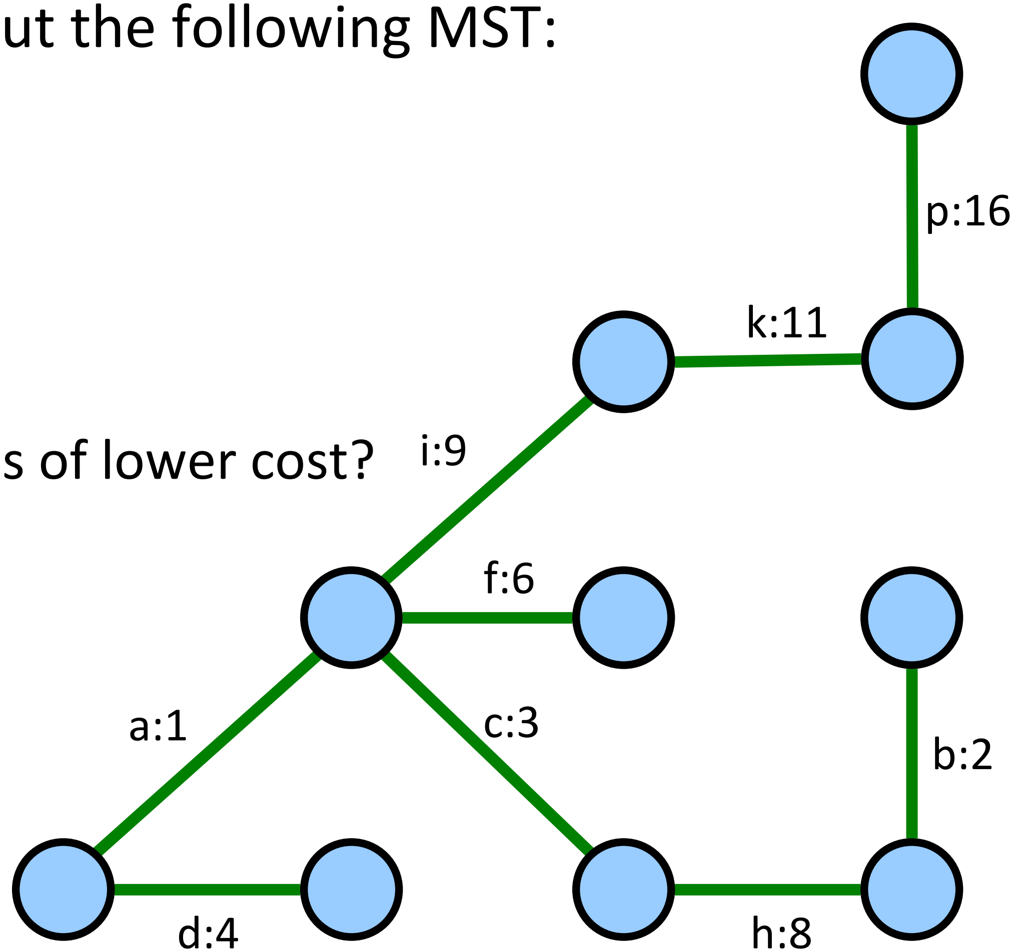


  - pq = { }

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
  - Can you find any spanning trees of lower cost? Of equal cost?



Implementing Kruskal

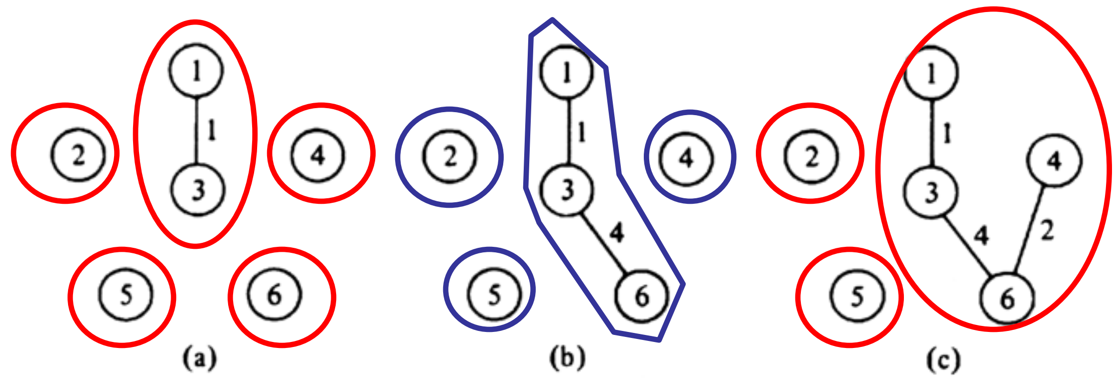
- What data structures should we use to implement this algorithm?

```
function kruskal(graph):
  Start with an empty structure for the MST
  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 MST.
    Otherwise, skip the edge.
```

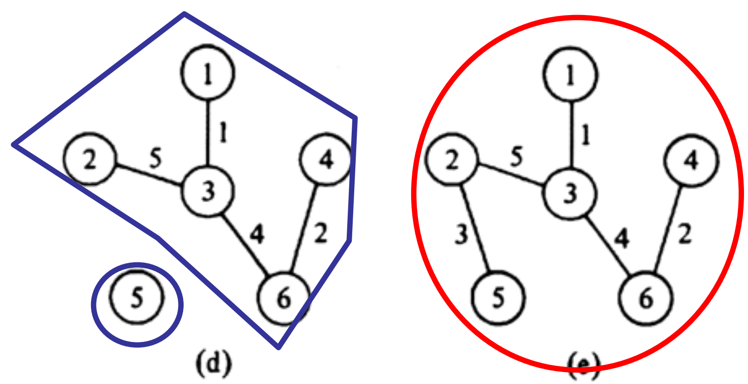
Vertex clusters

- 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.



Kruskal's Code

- How would we code Kruskal's algorithm to find a minimum spanning tree?
- What type of graph (adjacency list, adjacency matrix, or edge list) should we use?