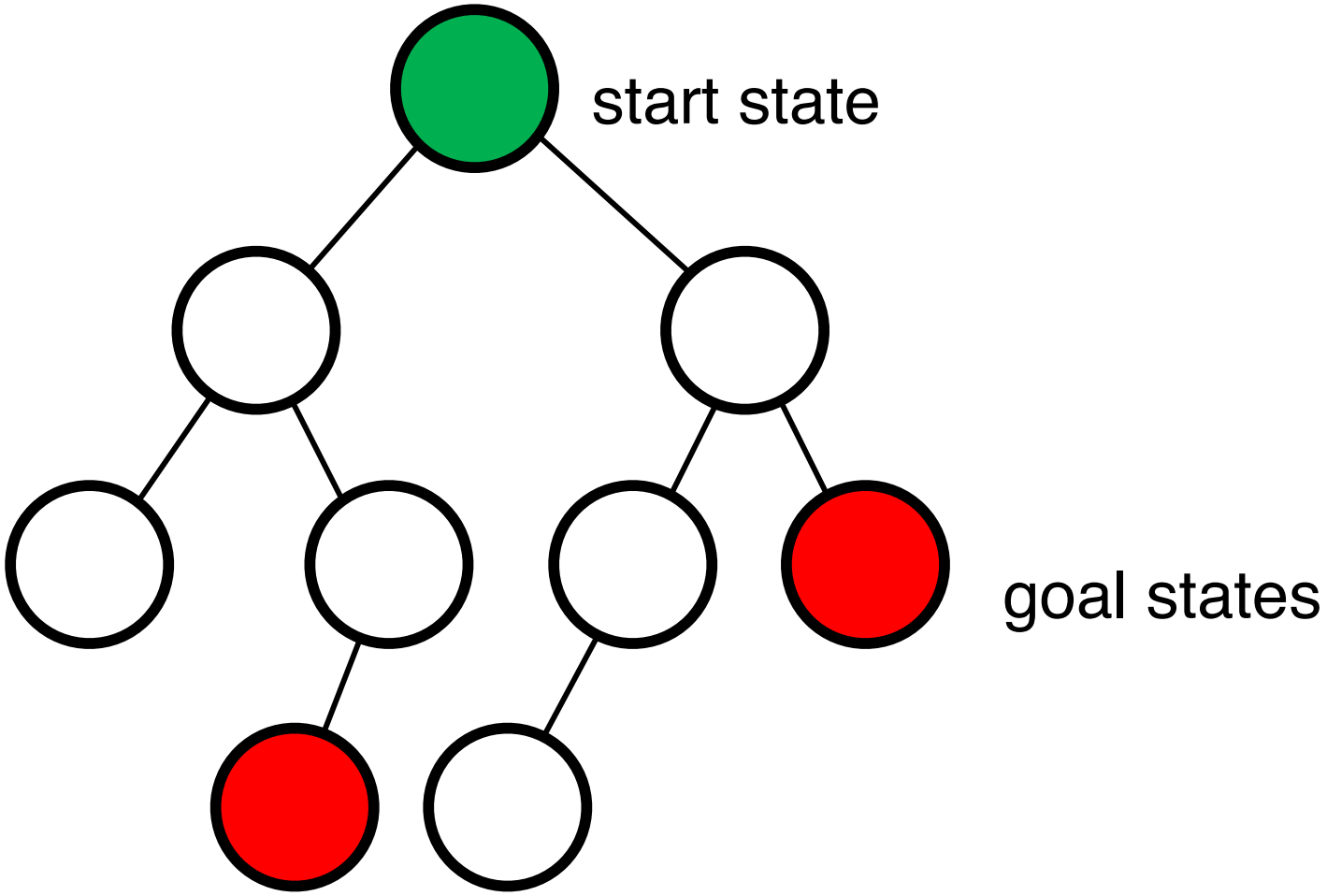
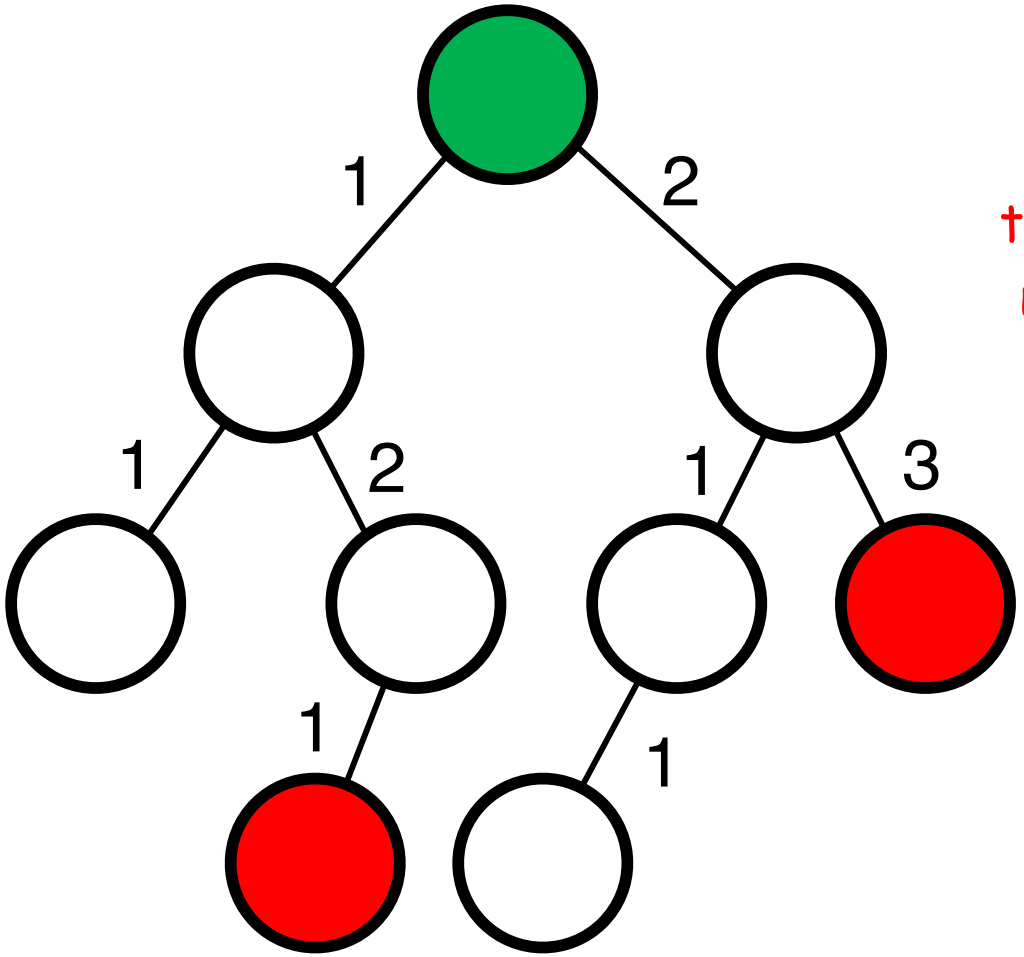
Shortest-Path Algorithms

Last Time



What If Edges Have Costs?



(for now we assume that the weights are **non-negative**)

What If Edges Have Costs?

Use a (minimum) **priority queue** to store the next nodes to visit

- cost = length of path from start to node

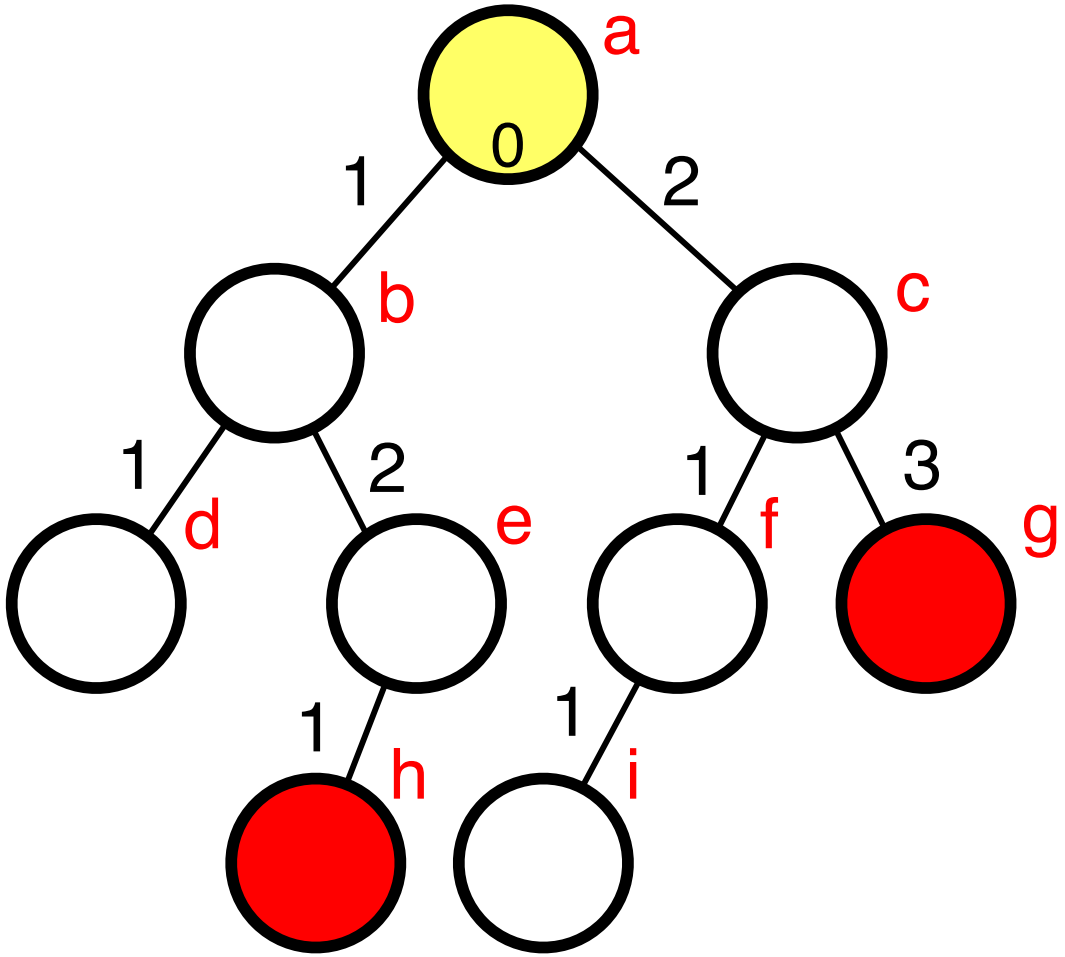
What If Edges Have Costs?

Use a (minimum) **priority queue** to store the next nodes to visit

- cost = length of path from start to node

This guarantees that nodes will be visited in order from closest to farthest

What If Edges Have Costs?



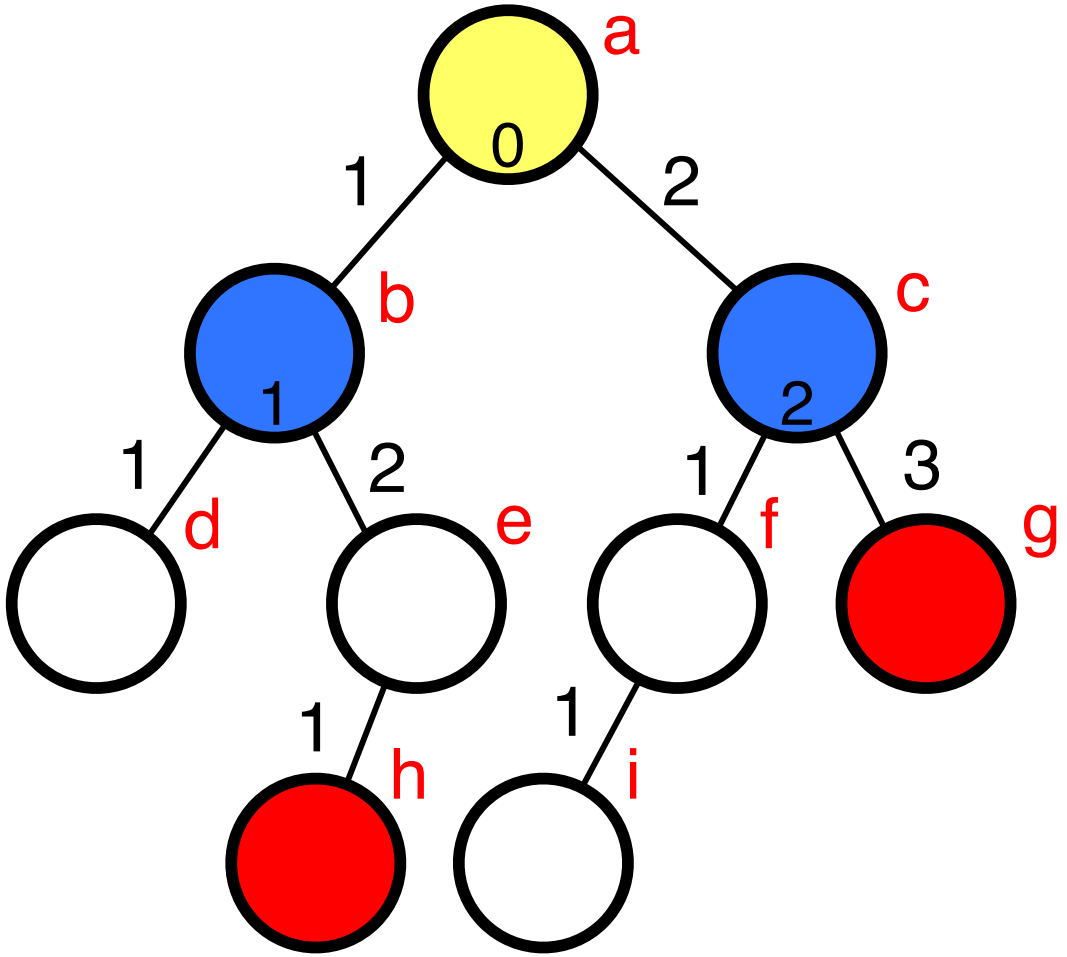
goal states
in queue
visited
current

Queue contents:

(a, 0)

What If Edges Have Costs?

goal states
in queue
visited
current

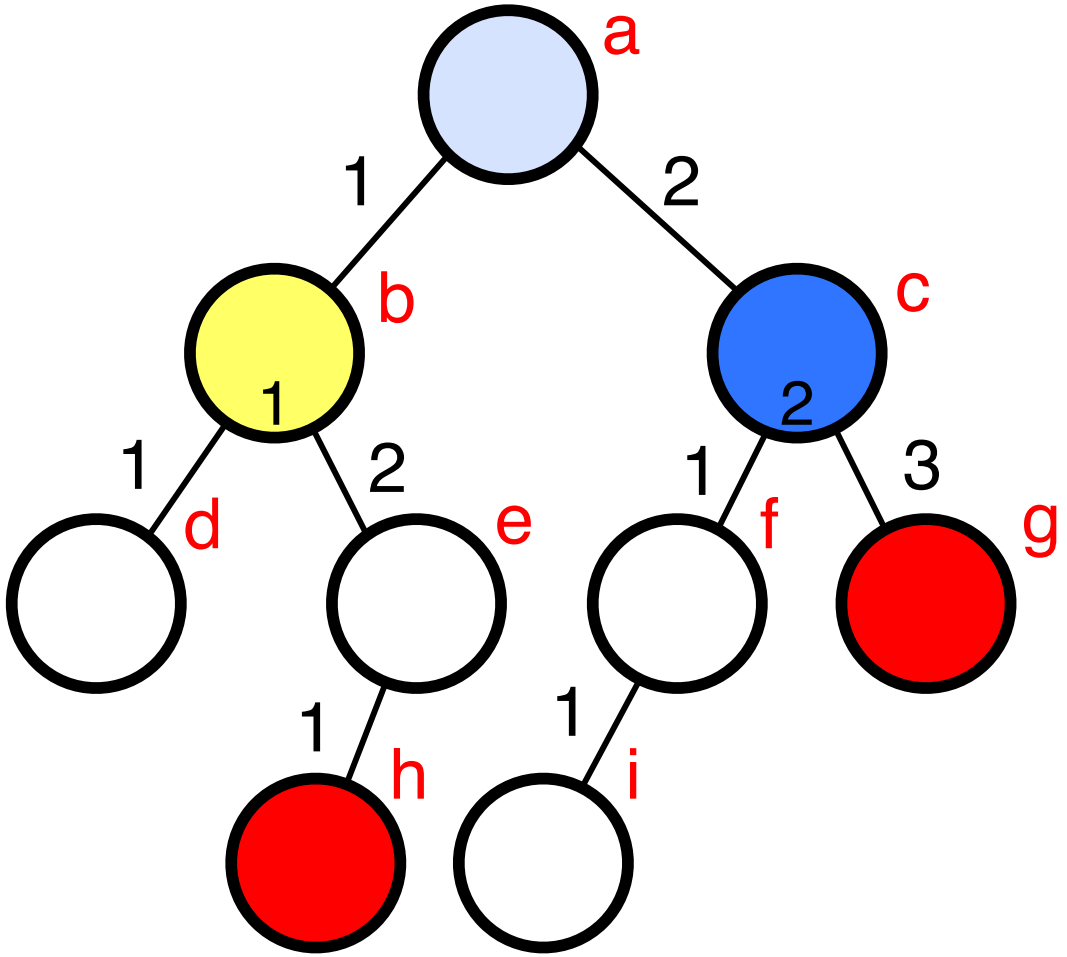


Queue contents:

(a, 0)
(b, 1)
(c, 2)

What If Edges Have Costs?

goal states
in queue
visited
current



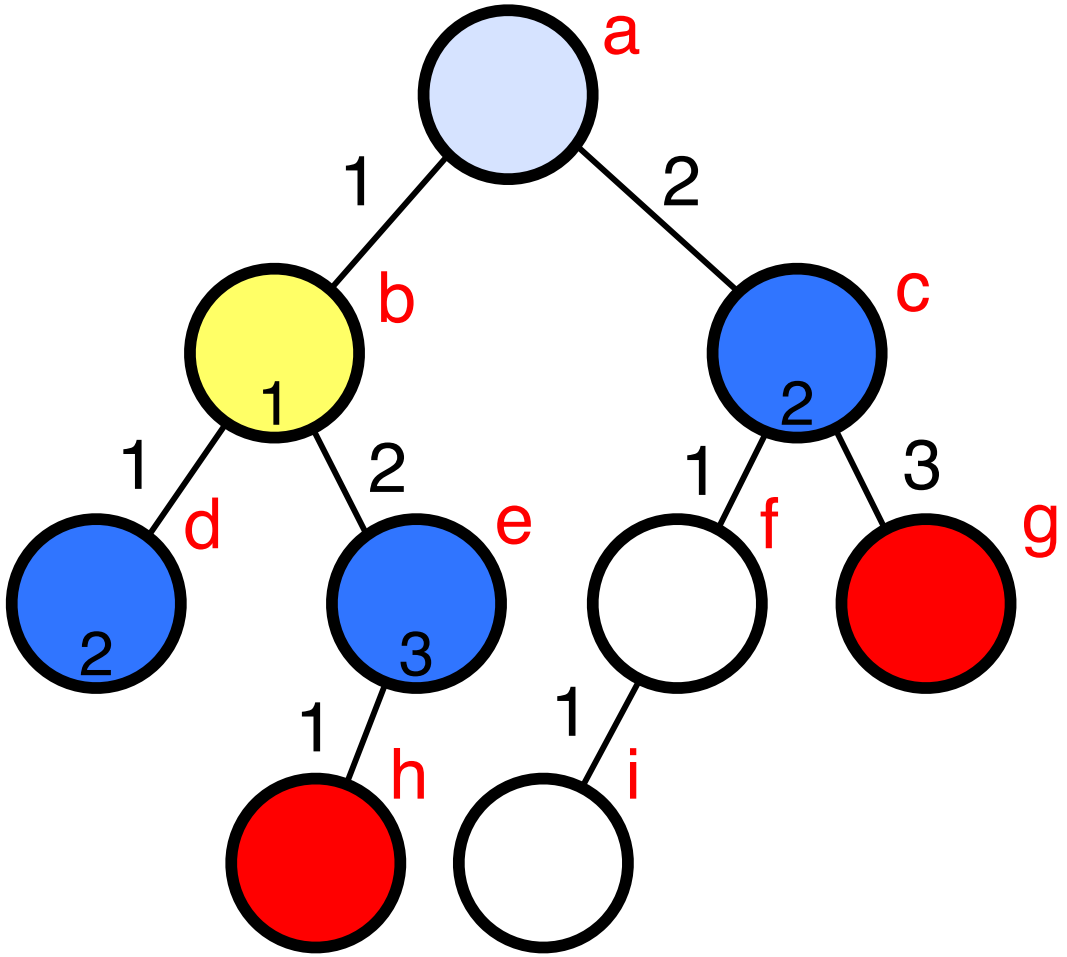
Queue contents:

(b, 1)
(c, 2)

Invariant: visited paths are shorter than unvisited paths

What If Edges Have Costs?

goal states
in queue
visited
current

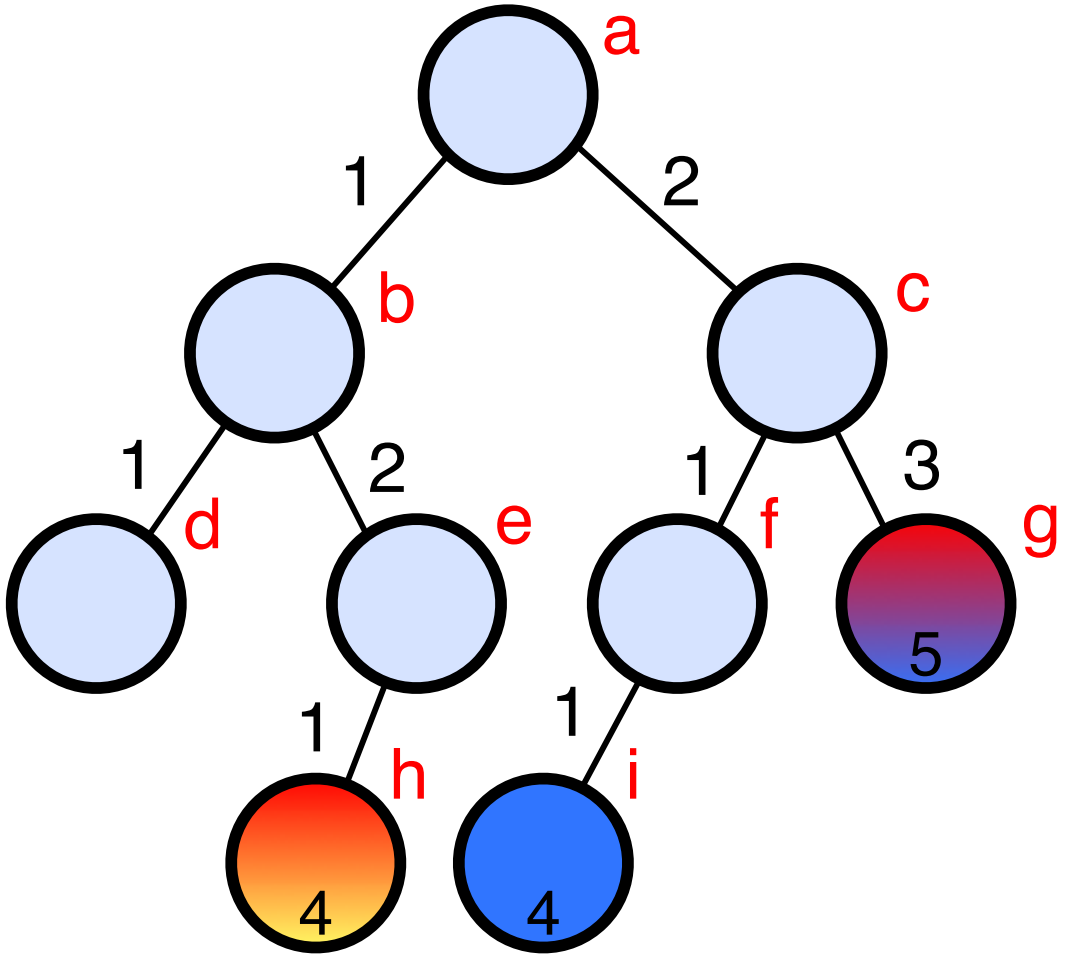


Queue contents:

(b, 1)
(d, 2)
(c, 2)
(e, 3)

Invariant: visited paths are shorter than unvisited paths

What If Edges Have Costs?



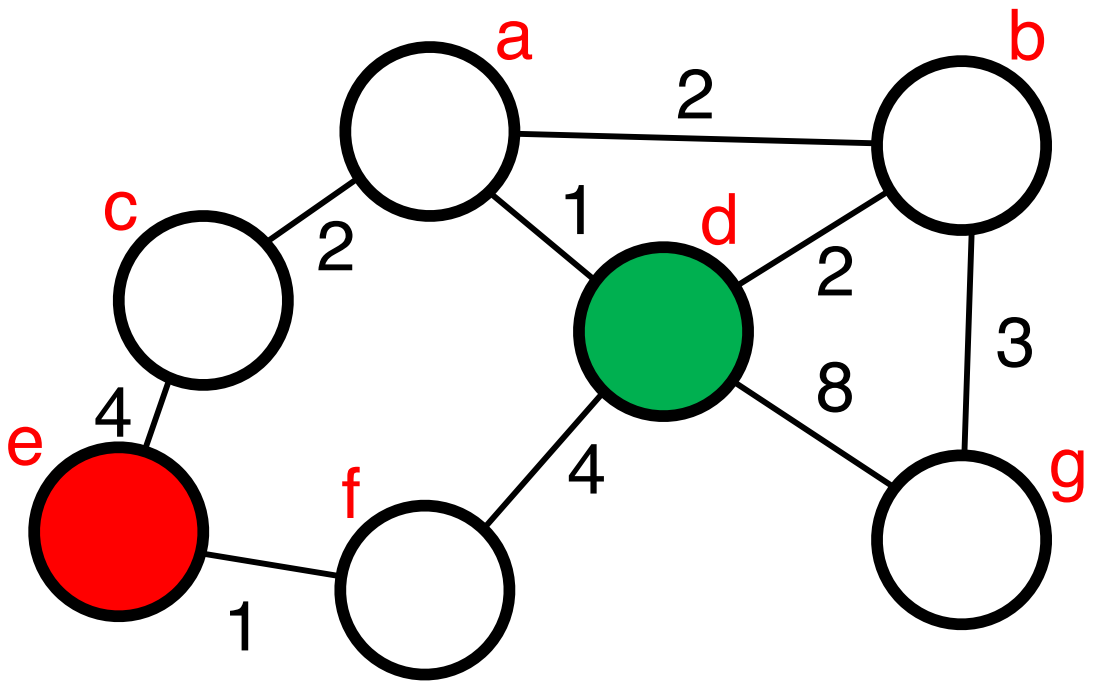
Queue contents:

(h, 4)
(i, 4)
(g, 5)

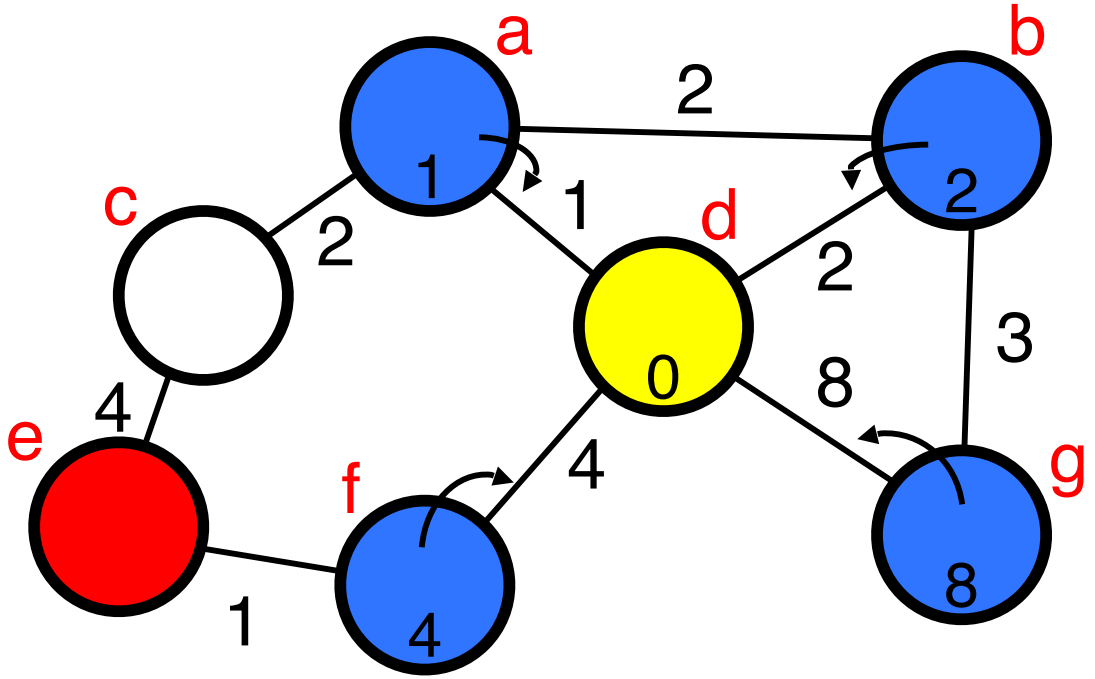
Invariant: visited paths are shorter than unvisited paths

start state
goal states

General Graphs



Dijkstra's Algorithm



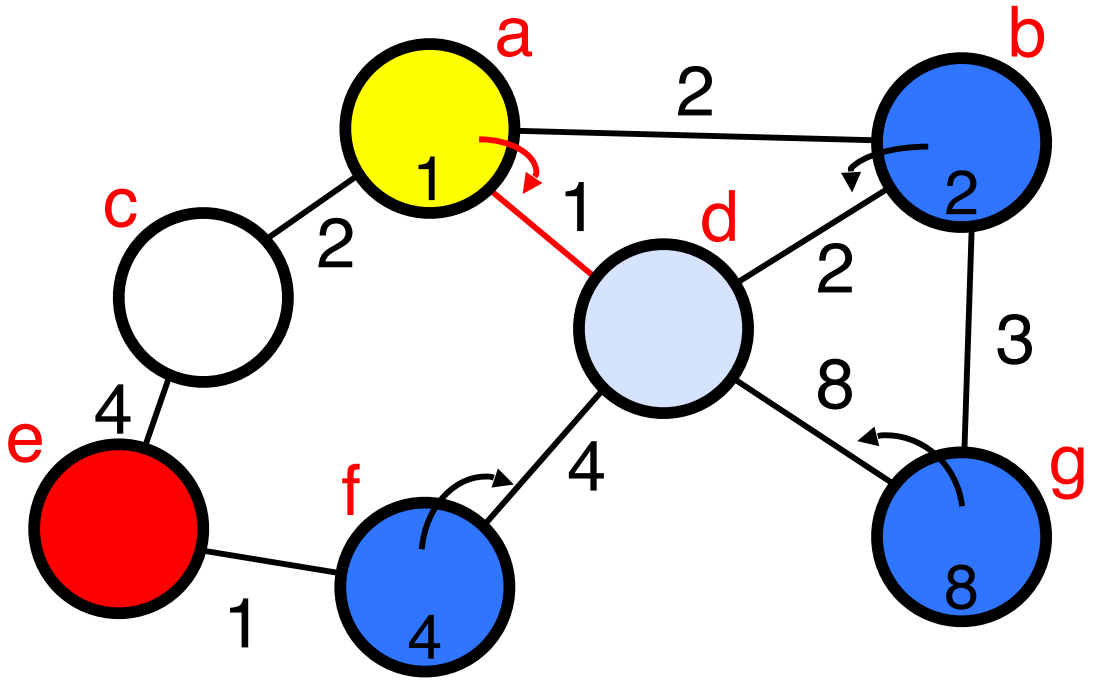
start state
goal states
in queue
visited
current

Queue contents:

(d, 0, none)
(a, 1, d)
(b, 2, d)
(f, 4, d)
(g, 8, d)

Dijkstra's Algorithm

start state
goal states
in queue
visited
current

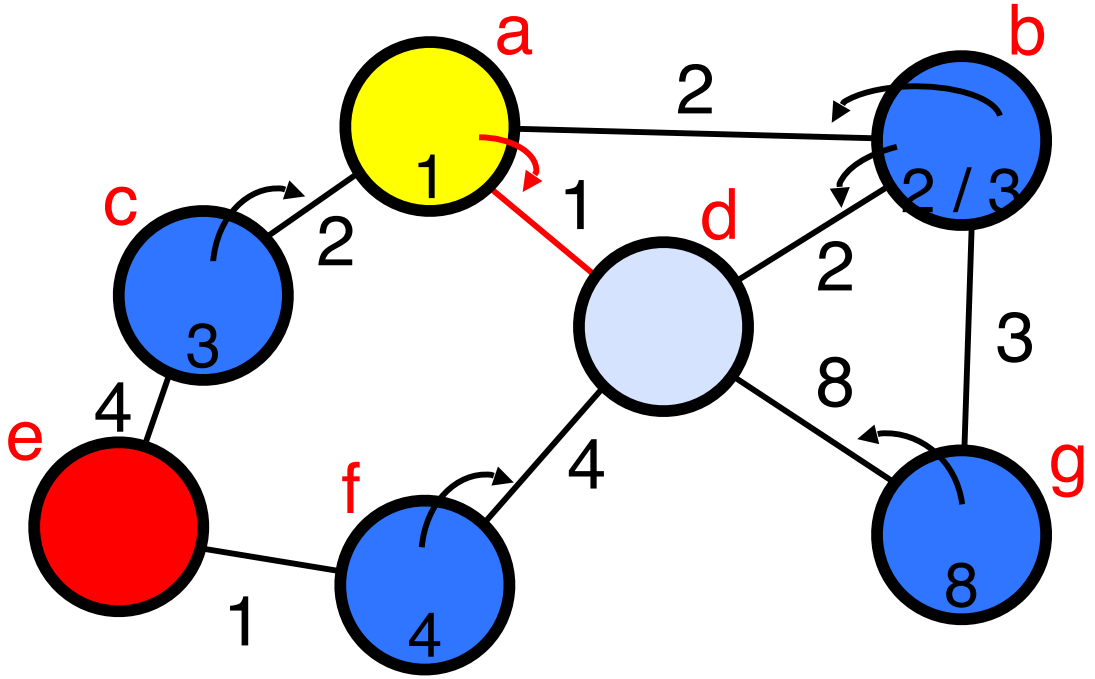


Queue contents:

(a, 1, d)
(b, 2, d)
(f, 4, d)
(g, 8, d)

Invariant: red paths are shortest possible

Dijkstra's Algorithm



start state
goal states
in queue
visited
current

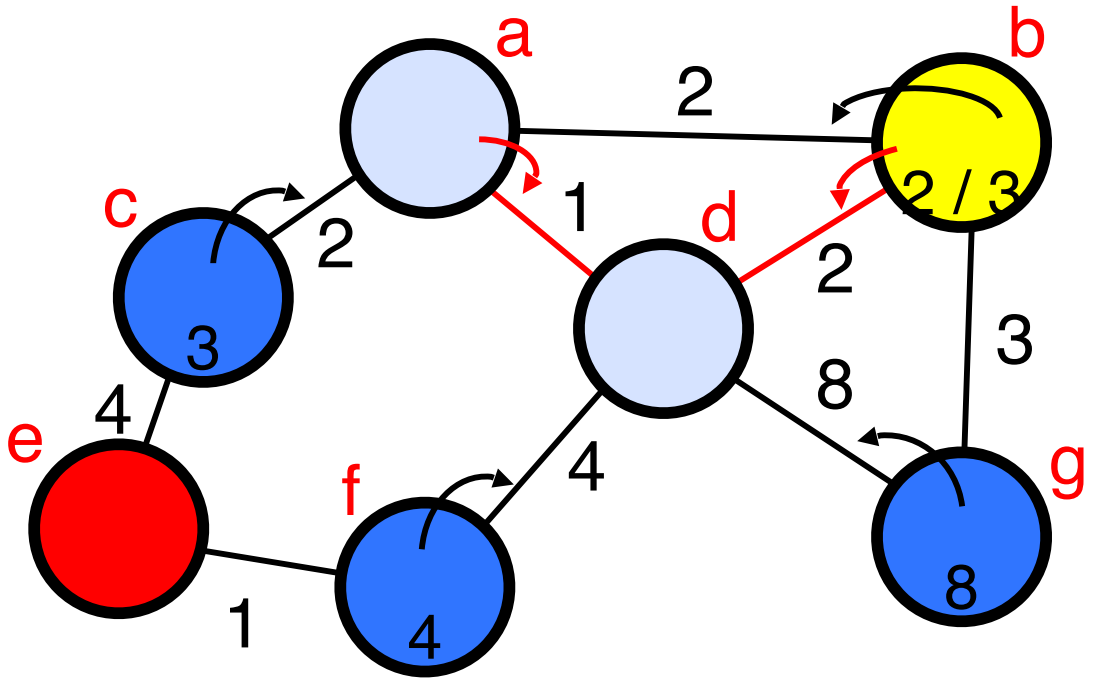
Queue contents:

(b, 2, d)
(b, 3, a)
(c, 3, a)
(f, 4, d)
(g, 8, d)

Invariant: red paths are shortest possible

Dijkstra's Algorithm

start state
goal states
in queue
visited
current



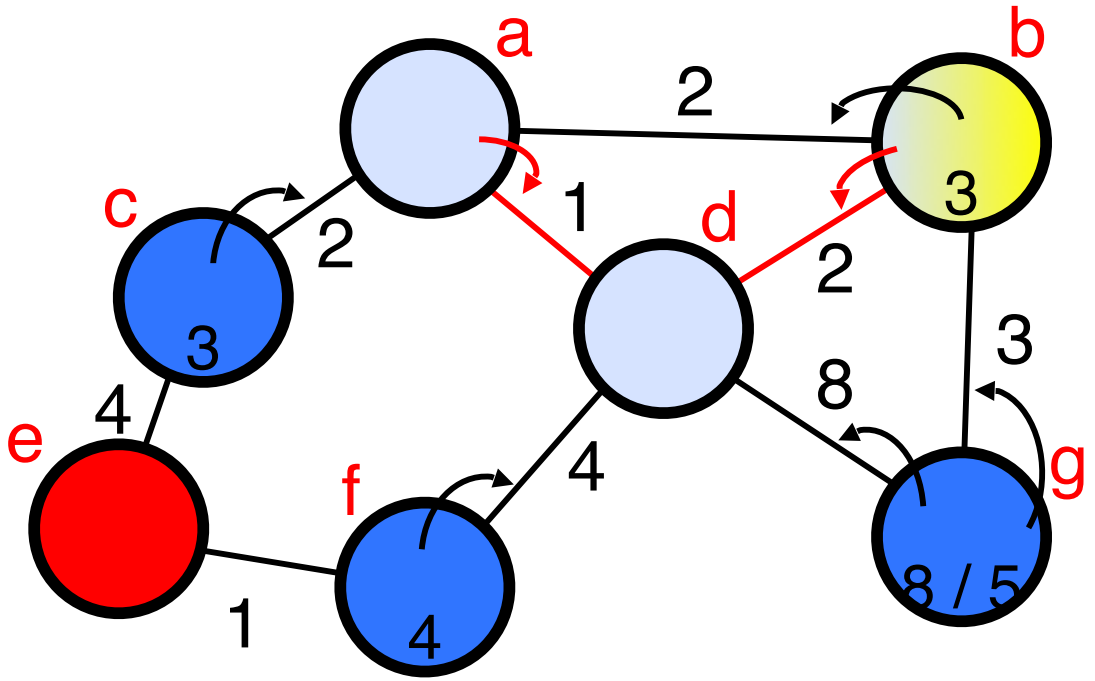
Queue contents:

(b, 2, d)
(b, 3, a)
(c, 3, a)
(f, 4, d)
(g, 8, d)

Invariant: red paths are shortest possible

Dijkstra's Algorithm

start state
goal states
in queue
visited
current



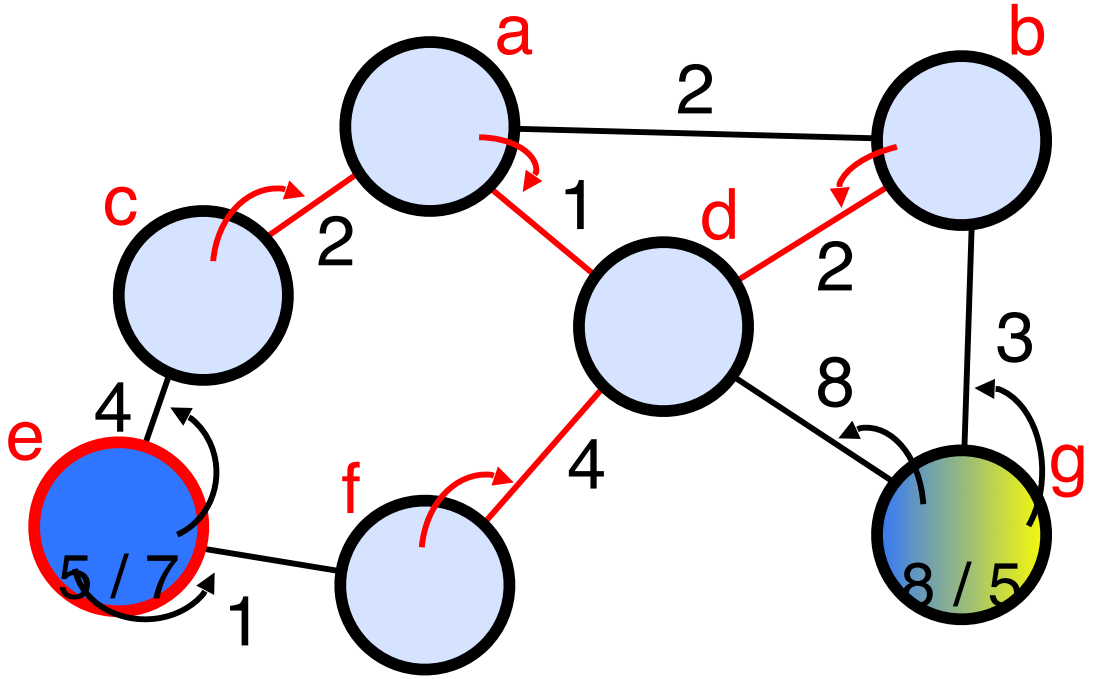
Queue contents:

(b, 3, a)
(c, 3, a)
(f, 4, d)
(g, 5, b)
(g, 8, d)

Invariant: red paths are shortest possible

Dijkstra's Algorithm

start state
goal states
in queue
visited
current



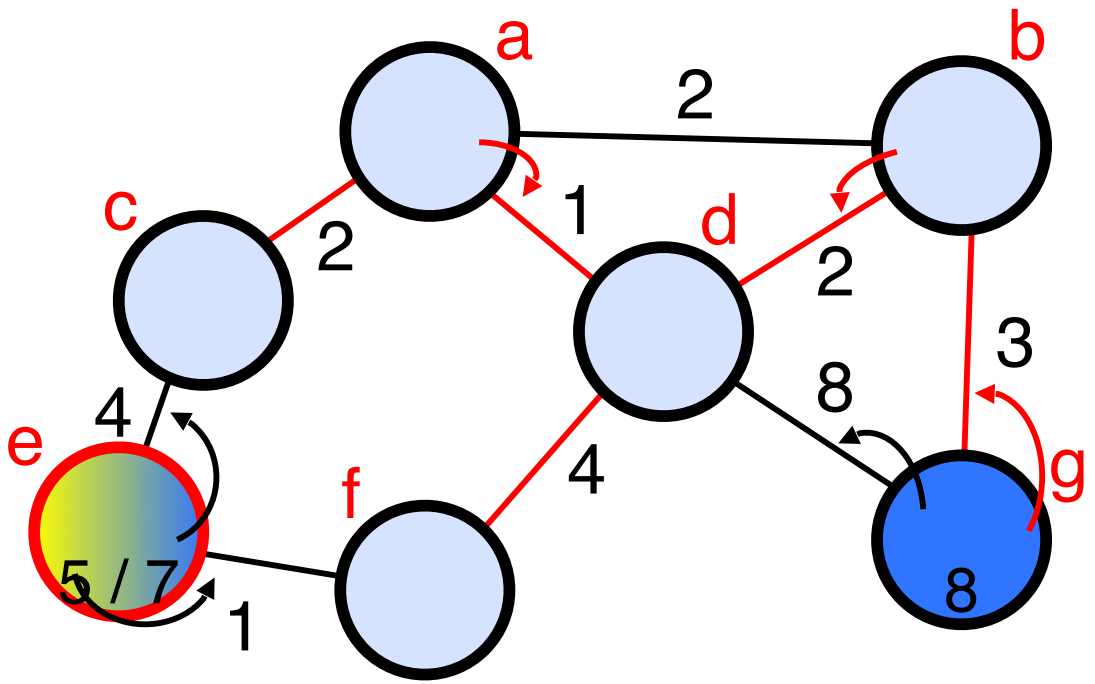
Queue contents:

(g, 5, b)
(e, 5, f)
(e, 7, c)
(g, 8, d)

Invariant: red paths are shortest possible

Dijkstra's Algorithm

start state
goal states
in queue
visited
current

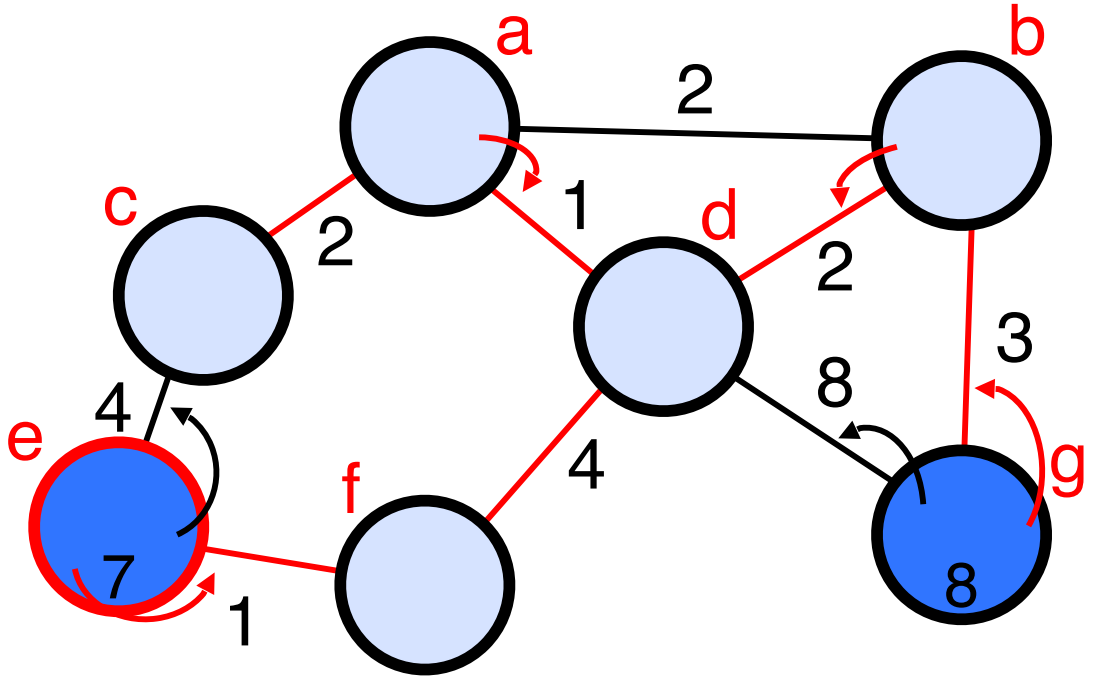


Queue contents:

(e, 5, f)
(e, 7, c)
(g, 8, d)

Invariant: red paths are shortest possible

Dijkstra's Algorithm



start state
goal states
in queue
visited
current

Queue contents:

(e, 7, c)
(g, 8, d)

solution: cost 5,
d -> f -> e

Invariant: red paths are shortest possible

Dijkstra's Algorithm

```
dijkstra(A,B)
for each vertex v:
    v.visited = false;
p_queue Q = {(A,0)};
```

```
while(!Q.empty())
  v = Q.pop();
  if(v.node == B) return v.cost;
  if(v.node.visited) continue;
  v.node.visited = true;
  for each neighbor w:
    if(!w.visited)
      Q.push((w, v.dist + d_vw));
return infinity;
```

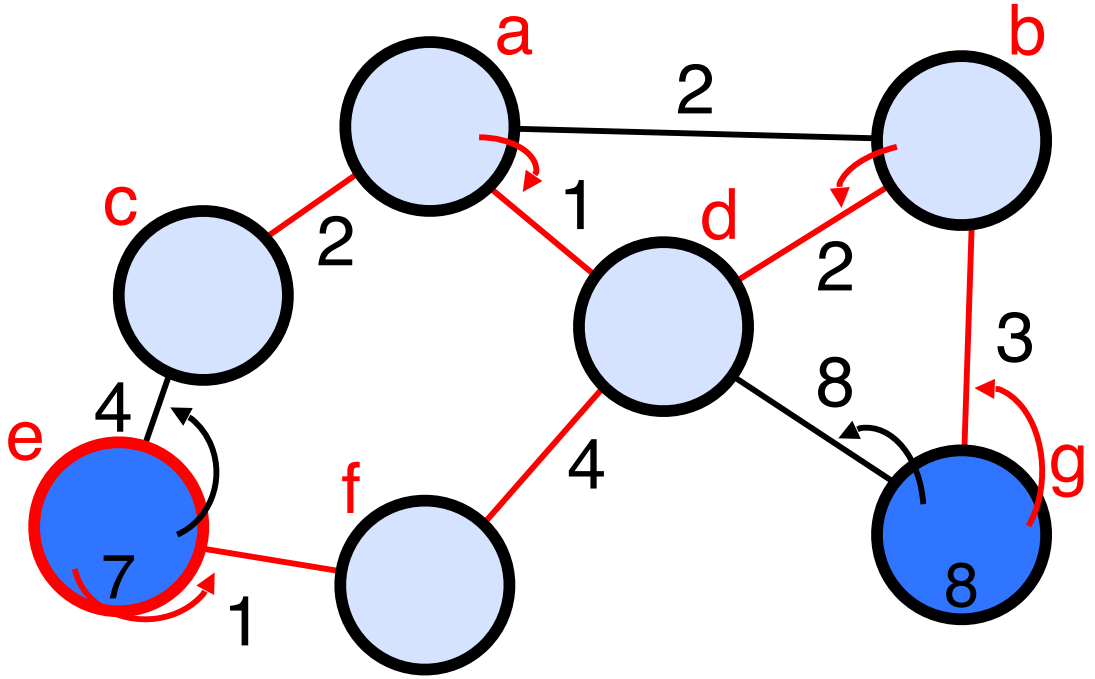
Dijkstra's Algorithm w/ Path

```
dijkstra(A,B)
for each vertex v:
     v.visited = false;
    v.prev = -1;
p_queue Q = {(A,0,-1)};
```

```
while(!Q.empty())
  v = Q.pop();
  if(v.node == B) return v.cost;
  if(v.node.visited) continue;
  v.node.visited = true;
  v.node.prev = v.prev;
  for each neighbor w:
    if(!w.visited)
      Q.push((w, v.dist + d_vw,
                    v.node));
return infinity;
```

Dijkstra's Algorithm

start state
goal states
in queue
visited
current



Queue contents:

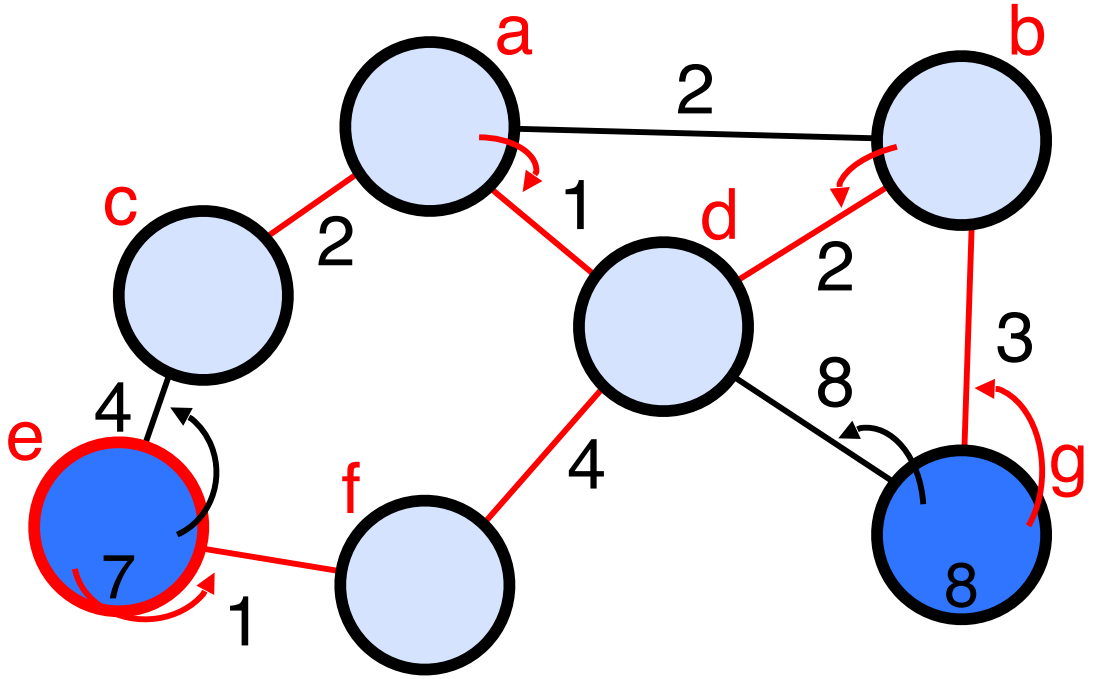
(e, 7, c)
(g, 8, d)

solution: cost 5,
d -> f -> e

What's the run time?

Dijkstra's Algorithm

start state
goal states
in queue
visited
current



Queue contents:

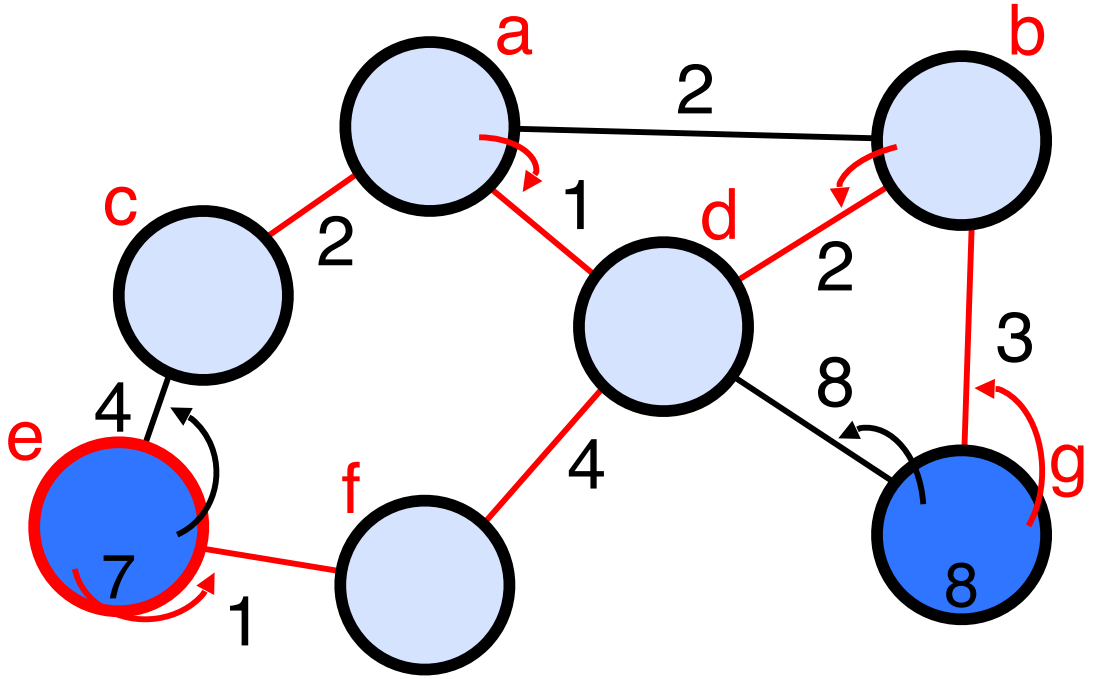
(e, 7, c)
(g, 8, d)

solution: cost 5,
d -> f -> e

What's the run time? $O(|E| \log |E|)$

Dijkstra's Algorithm

start state
goal states
in queue
visited
current



Queue contents:

(e, 7, c)
(g, 8, d)

solution: cost 5,
d -> f -> e

What's the run time? $O(|E| \log |E|)$

(can be improved to $O(|E| + |V| \log |V|)$)