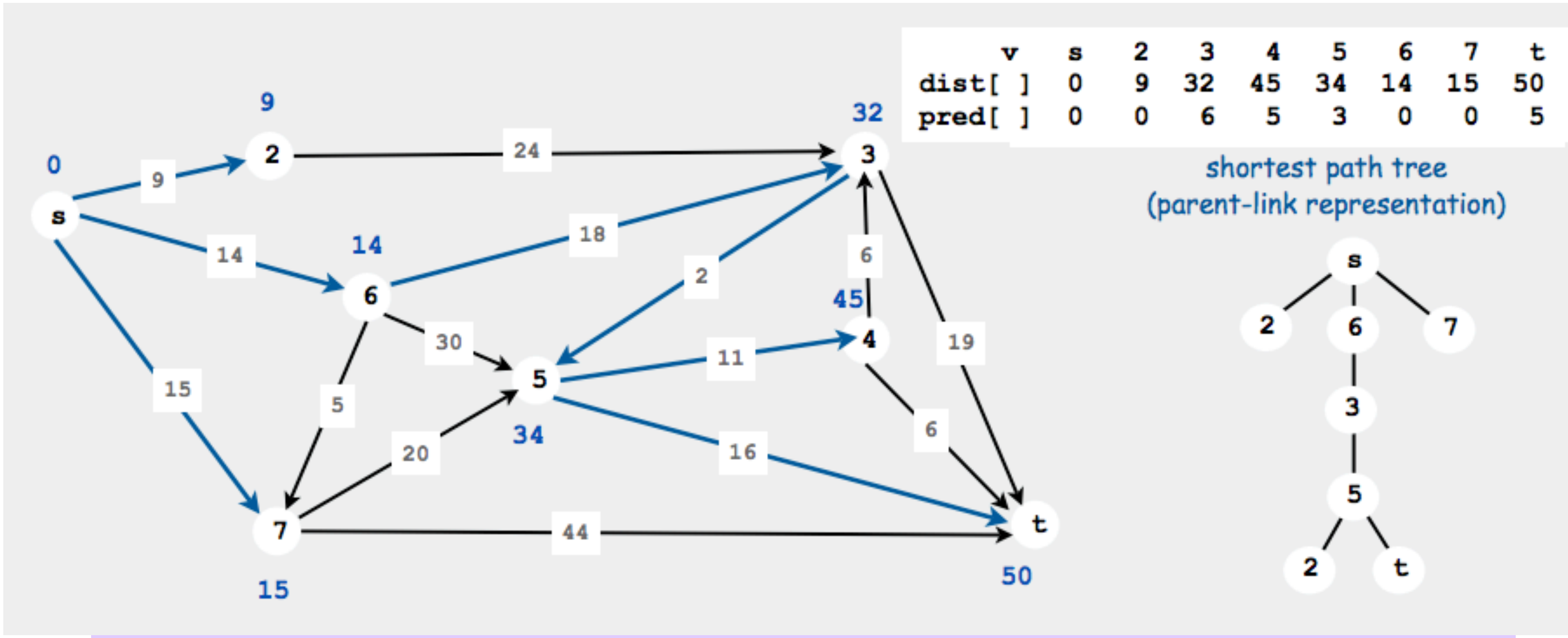
Single Source Shortest Path

Find distance (and shortest path) from s to every other vertex in a weighted graph.

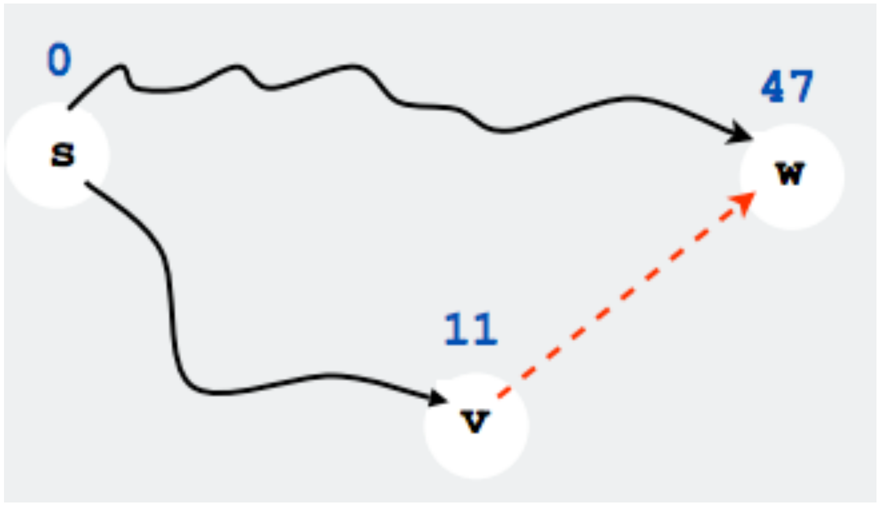


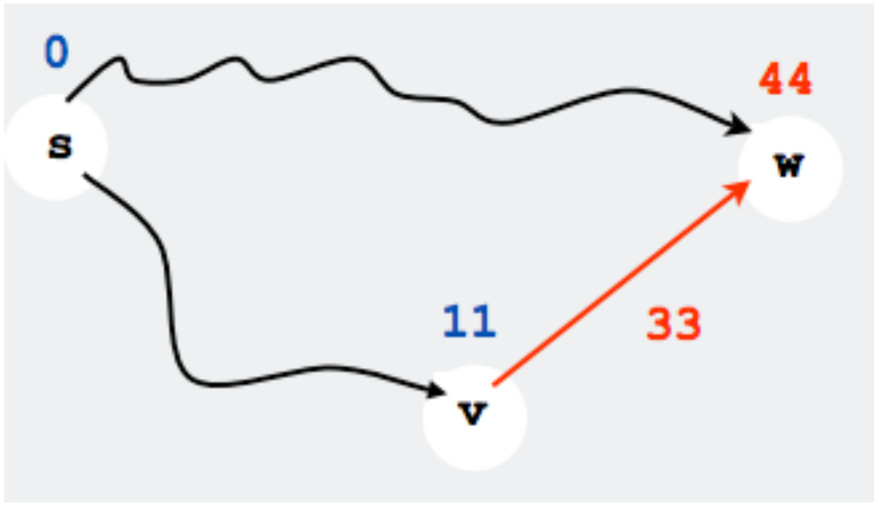
v	s	2	3	4	5	6	7	t
dist[ ]	0	9	32	45	34	14	15	50
pred[ ]	0	0	6	5	3	0	0	5

Why must we assume that the graph has no cycles with a negative weight?

Edge Relaxation

For all v, dist[v] is the length of some path from s to v.
Relaxation along edge e from v to w:
Relaxation sets dist[w] to the length of a shorter path from s to w (if v-w gives one)



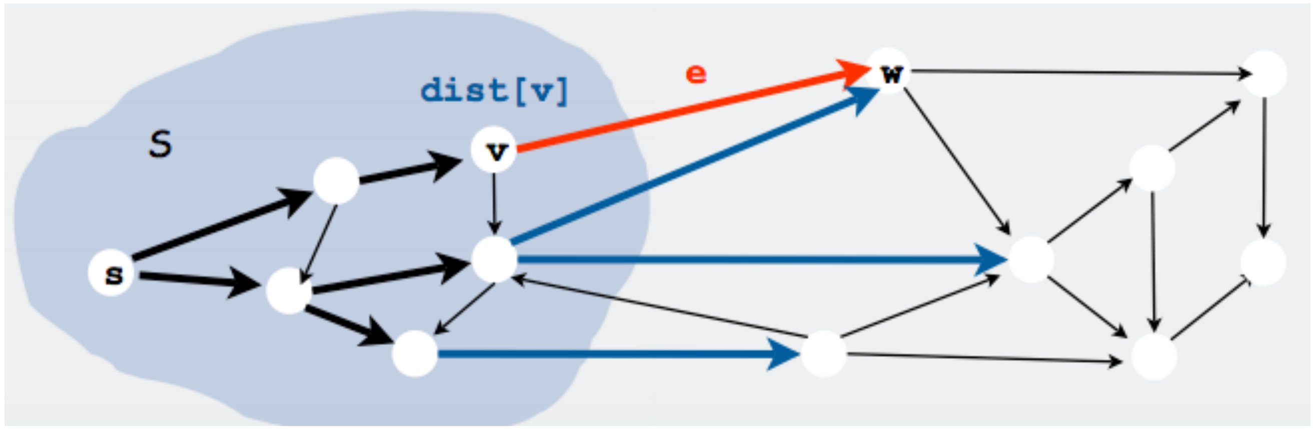


Write the code which will change dist and pred if needed

Dijkstra's algorithm

S: set of vertices for which the shortest path length from s is known.

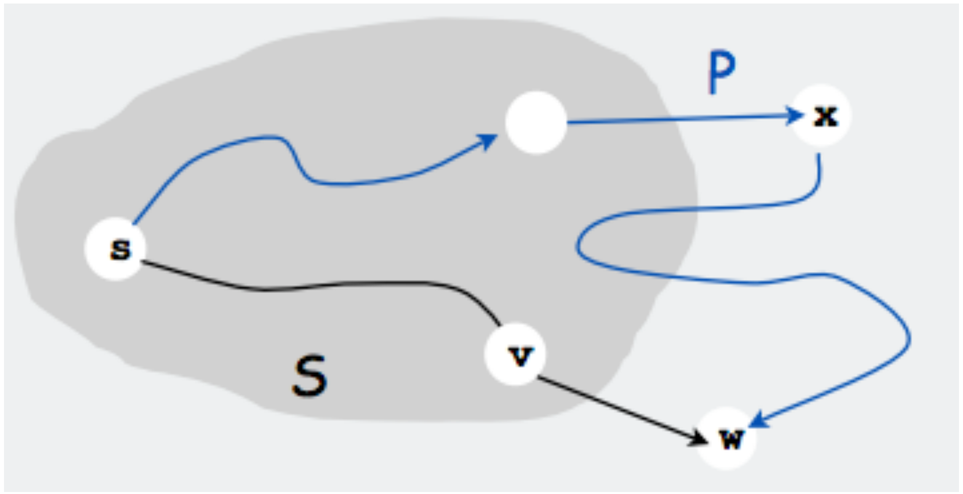
```
Initialize S to s, dist[s] to 0, dist[v] to ∞ for all other v
Repeat until S contains all vertices connected to s
    find e with v in S and w not in S that minimizes dist[v] + e.weight()
    relax along that edge
    add w to S
```



Why do you think w can be added to S?

Dijkstra's algorithm proof of correctness

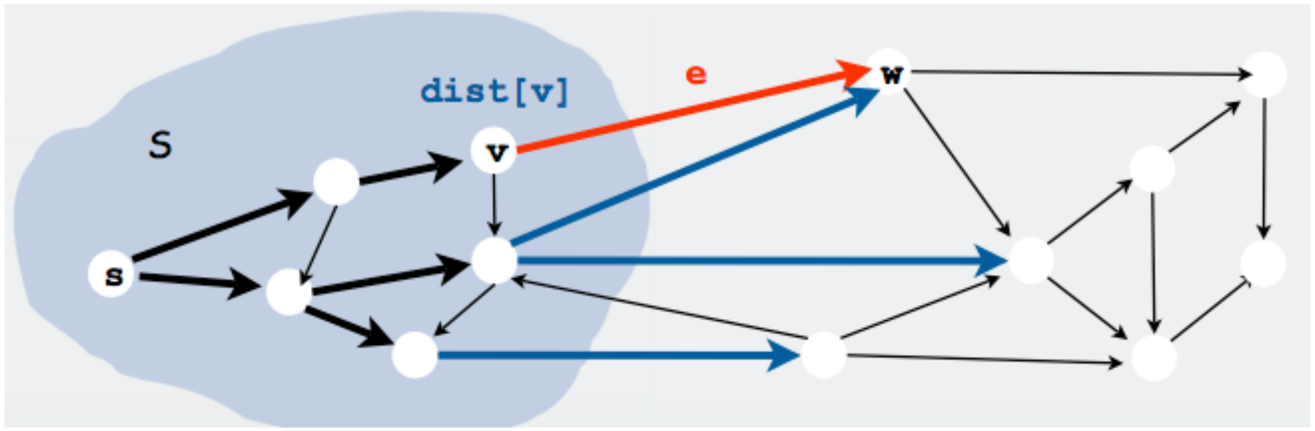
Proof. (by induction on size of S)
Let w be next vertex added to S.
Let P* be the s-w path through v.
Consider any other s-w path P, and let x be first node on path outside S.



Why is the path P* shorter than the path P?

Dijkstra's Implementation and Cost

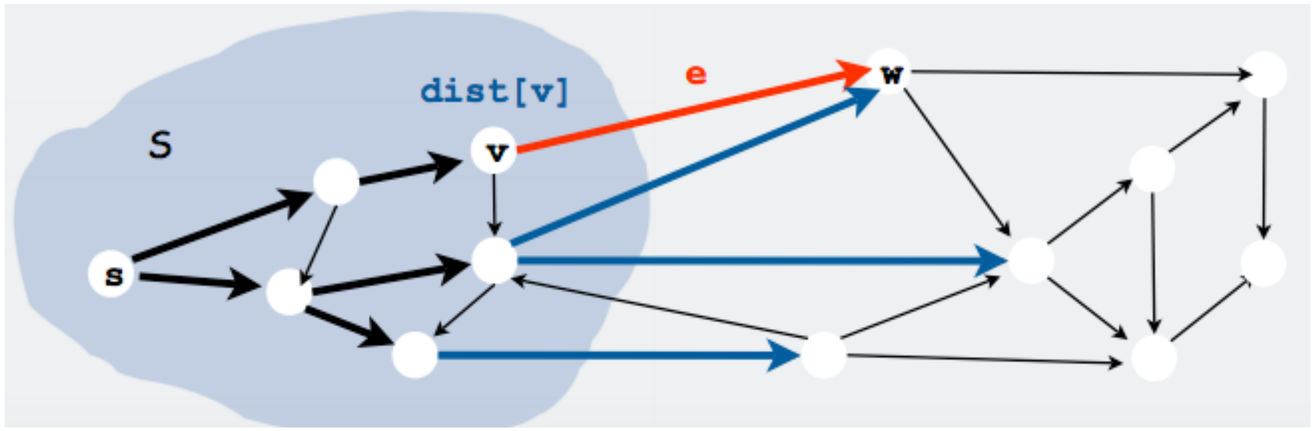
```
Initialize S to s, dist[s] to 0, dist[v] to ∞ for all other v
Repeat until S contains all vertices connected to s
    find e with v in S and w not in S that minimizes dist[v] + e.weight()
    relax along that edge
    add w to S
```



What data structure should we use to maintain dist? Hint: We need to extract the minimum.

Dijkstra's Implementation and Cost

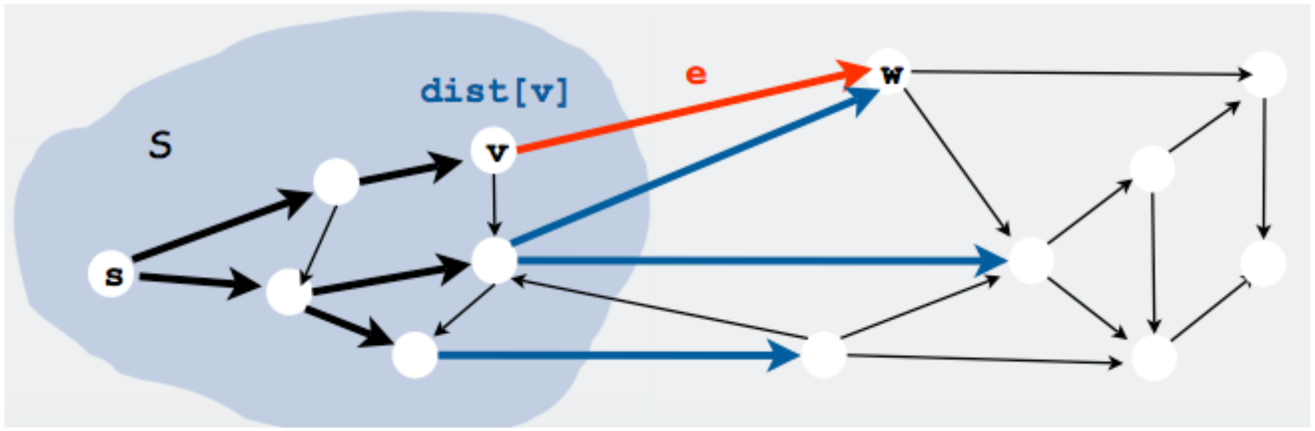
```
Initialize S to s, dist[s] to 0, dist[v] to ∞ for all other v
Repeat until S contains all vertices connected to s
    find e with v in S and w not in S that minimizes dist[v] + e.weight()
    relax along that edge
    add w to S
```



How many times is each edge examined (and relaxed)?

Dijkstra's Implementation and Cost

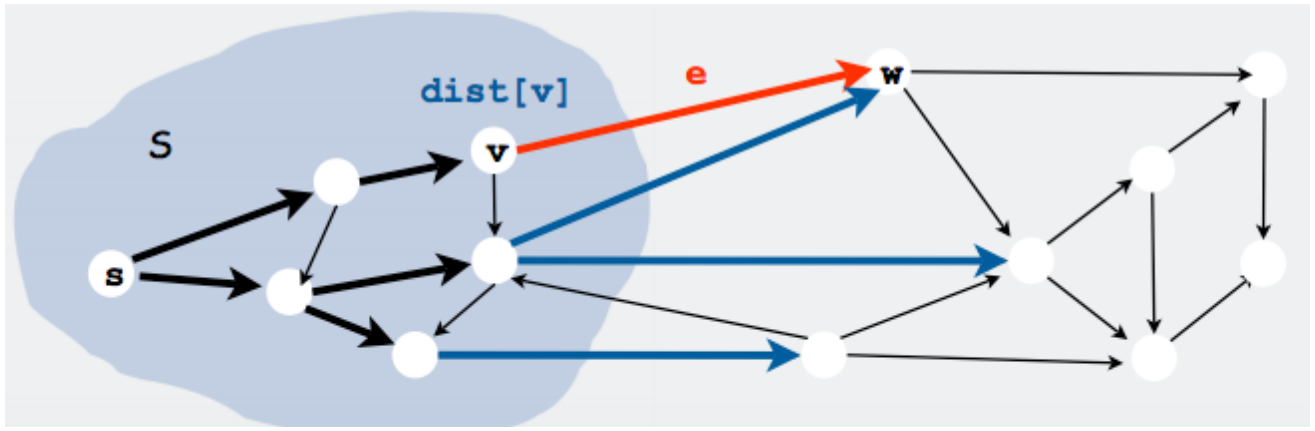
```
Initialize S to s, dist[s] to 0, dist[v] to ∞ for all other v
Repeat until S contains all vertices connected to s
    find e with v in S and w not in S that minimizes dist[v] + e.weight()
    relax along that edge
    add w to S
```



What is the cost of extractMin (if using a heap)?

Dijkstra's Implementation and Cost

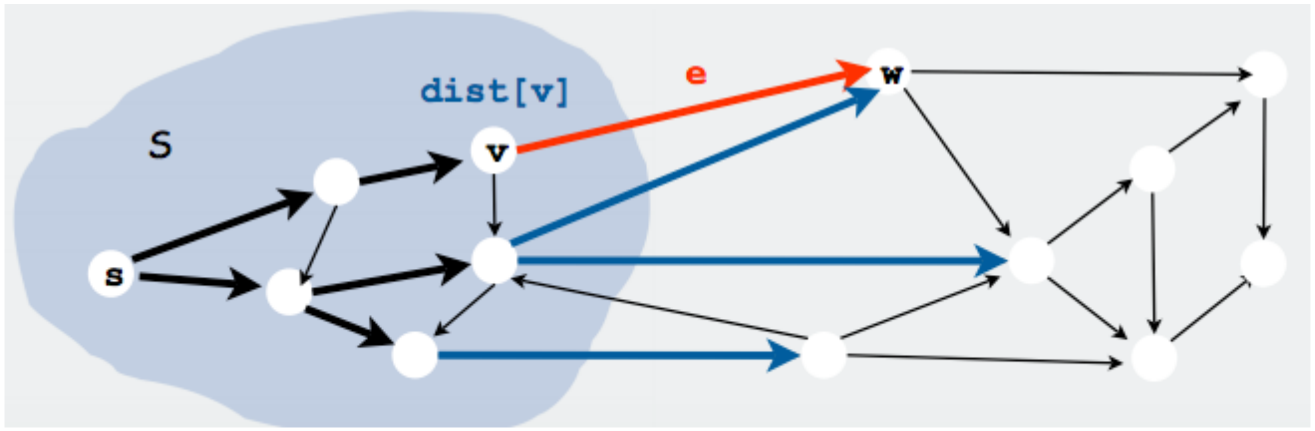
```
Initialize S to s, dist[s] to 0, dist[v] to ∞ for all other v
Repeat until S contains all vertices connected to s
    find e with v in S and w not in S that minimizes dist[v] + e.weight()
    relax along that edge
    add w to S
```



When we relax an edge we need to adjust the heap. What is the cost of this?

Dijkstra's Implementation and Cost

```
Initialize S to s, dist[s] to 0, dist[v] to ∞ for all other v
Repeat until S contains all vertices connected to s
    find e with v in S and w not in S that minimizes dist[v] + e.weight()
    relax along that edge
    add w to S
```



Putting it all together, what is the total order of growth?

Graph Search algorithms

Insight: All of our graph-search methods are the same algorithm!
Maintain a set of explored vertices S Grow S by exploring edges with exactly one endpoint leaving S.

DFS. Take edge from vertex which was discovered most recently.
BFS. Take from vertex which was discovered least recently.
Prim. Take edge of minimum weight.
Dijkstra. Take edge to vertex that is closest to s.

Bask in the glory of graph search!