Single Source Shortest Path: Dijkstra's Algorithm

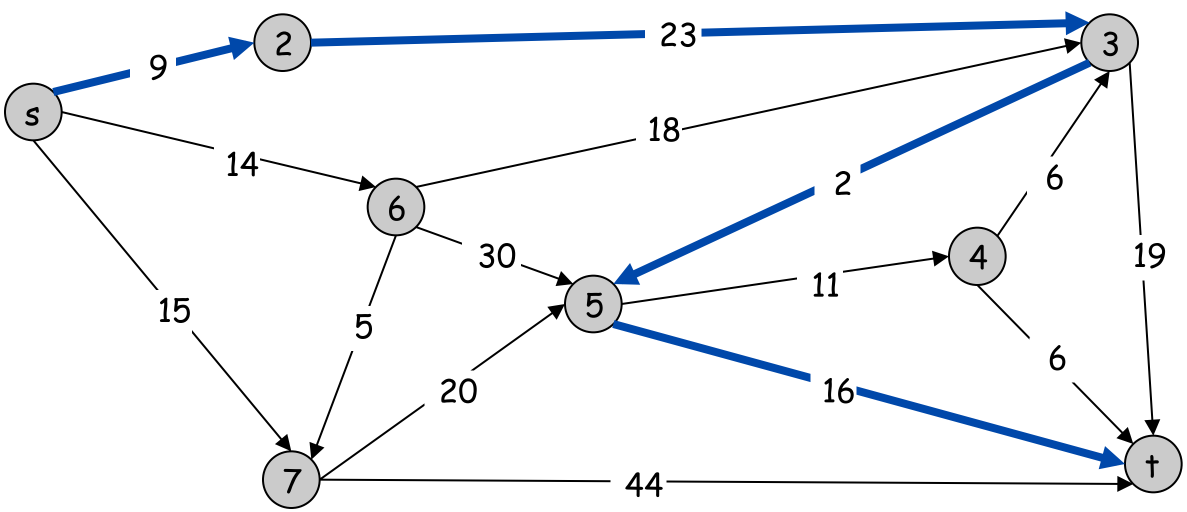
Shortest Path Problem

Shortest path network.

- Directed graph $G = (V, E, w)$.
- Source s, destination t.
- Weight w_e = weight of edge e.

Shortest path problem: find shortest directed path from s to t.

Total path weight = sum of edge weights in path



Path s-2-3-5-t
= 9 + 23 + 2 + 16
= 48.

Dijkstra's Algorithm

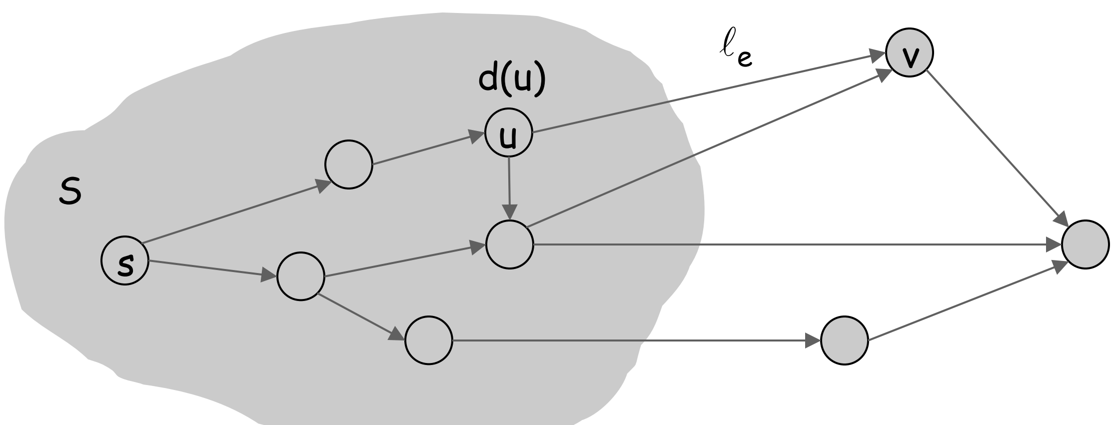
Dijkstra's algorithm.

- Maintain a set of explored nodes S for which we have determined the shortest path distance d(u) from s to u.
- Initialize S = { s }, d(s) = 0.
- Repeatedly choose unexplored node v which minimizes

$$\pi(v) = \min_{e = (u,v) : u \in S} d(u) + w_e ,$$

add v to S, and set d(v) = π(v).

shortest path to some u in explored part, followed by a single edge (u, v)



NOTE: w_e are assumed non-negative

Dijkstra's Algorithm

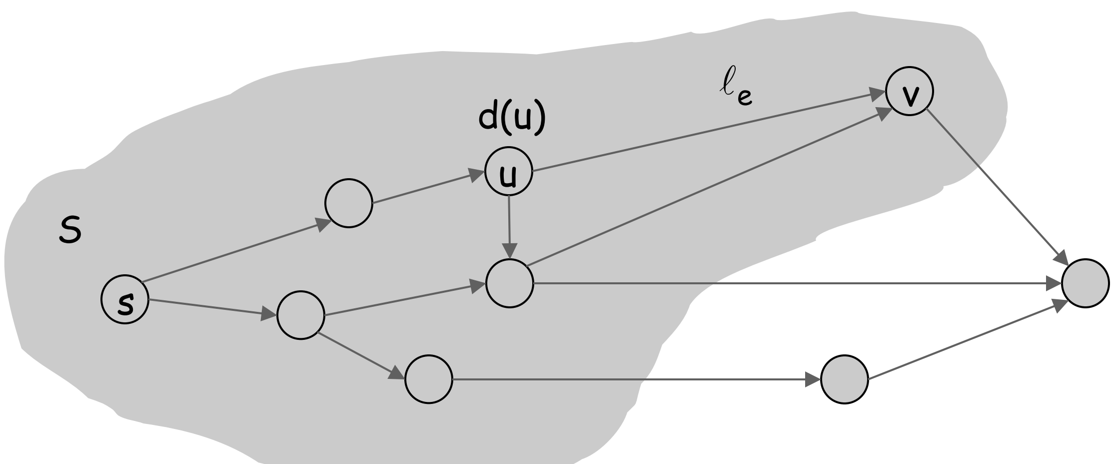
Dijkstra's algorithm.

- Maintain a set of explored nodes S for which we have determined the shortest path distance d(u) from s to u.
- Initialize S = { s }, d(s) = 0.
- Repeatedly choose unexplored node v which minimizes

$$\pi(v) = \min_{e = (u,v) : u \in S} d(u) + w_e ,$$

add v to S, and set d(v) = π(v).

shortest path to some u in explored part, followed by a single edge (u, v)



NOTE: w_e are assumed non-negative

Dijkstra's Algorithm

Dijkstra's algorithm.

- Can be specified for a single source, s, and single destination, t.
- However, each iteration determines the shortest path from s to some vertex v.

- Dijkstra's algorithm run for n iterations will find the shortest path from s to all vertices.

- Dijkstra works for directed as well as undirected graphs.
  - Requires non-negative weights.

- Dijkstra is a GREEDY algorithm!

Dijkstra's Algorithm

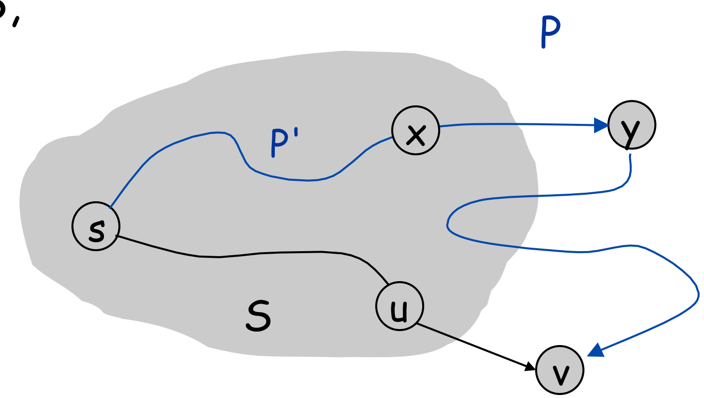
```
Dijkstra ( G=(V,E,w), s )
1. Let H = V - {s};
2. For every vertex v do
3.     dist[v]= ∞, parent[v]=null
4. dist[s]=0, parent[s] = none
5. Update (s)
6. For i=1 to n-1 do
7.     u=extract vertex from H
         of smallest weight
8.     Update(u)
• Return dist[]
```

```
Update (u)
1. For every neighbor v of u (such that v in H)
2.     If dist[v]>dist[u]+w(u,v) then
3.         dist[v]=dist[u]+w(u,v)
4.         parent[v] = u
```

Dijkstra's Algorithm: Proof of Correctness

Invariant. For each node $u \in S$, $d(u)$ is the length of the shortest s-u path.
Pf. (by induction on $|S|$)
Base case: $|S| = 1$ is trivial.
Inductive hypothesis: Assume true for $|S| = k \geq 1$.

- Let v be next node added to S, and let u-v be the chosen edge.
- The shortest s-u path plus (u, v) is an s-v path of length $\pi(v)$.
- Consider any s-v path P. We'll see that it's no shorter than $\pi(v)$.
- Let x-y be the first edge in P that leaves S, and let P' be the subpath to x.
- P is already too long as soon as it leaves S.



$$\underbrace{\ell(P) \geq \ell(P') + \ell(x,y)}_{\text{nonnegative weights}} \quad \underbrace{\geq d(x) + \ell(x, y)}_{\text{inductive hypothesis}} \quad \underbrace{\geq \pi(y)}_{\text{defn of } \pi(y)} \quad \underbrace{\geq \pi(v)}_{\text{Dijkstra chose v instead of y}}$$

Dijkstra – Complexity

Complexity Analysis for $G = (V,E)$, $|V| = n$, $|E| = m$

- We store minimum path weight information for each vertex in a heap data structure
  - Each heap operation requires $O(\log n)$ time

- n-1 iterations in which an EXTRACT_MIN heap operation is performed
  - $O(n \log n)$

- Each edge can result in at most one CHANGE_KEY heap operation
  - $O(m \log n)$

- Overall complexity: $O(m \log n)$

- (This is the same analysis as for Prim's MST algorithm)

Dijkstra – Complexity

Complexity Analysis for $G = (V,E)$, $|V| = n$, $|E| = m$

- Alternatively, store edge information in an adjacency matrix and store minimum path weight information for each vertex in a separate list

- n iterations in which the vertex with minimum path weight is chosen:
  - $O(n)$ per iteration, $O(n^2)$ total

- In each iteration, neighbors of the added vertex may have their minimum path weight updated:
  - $O(n)$ per iteration, $O(n^2)$ total

- Overall complexity: $O(n^2)$
  - Note: this is actually better than using a heap in the case that G is dense ($m \approx n^2$)

- (This is the same analysis is for Prim's MST algorithm)