Dijkstra's algorithm

Tufts University

Reminder: Single-Source Shortest Paths

Input: directed graph $G(E, V)$, $s \in V$, weight $w : E \to \mathbb{R}$

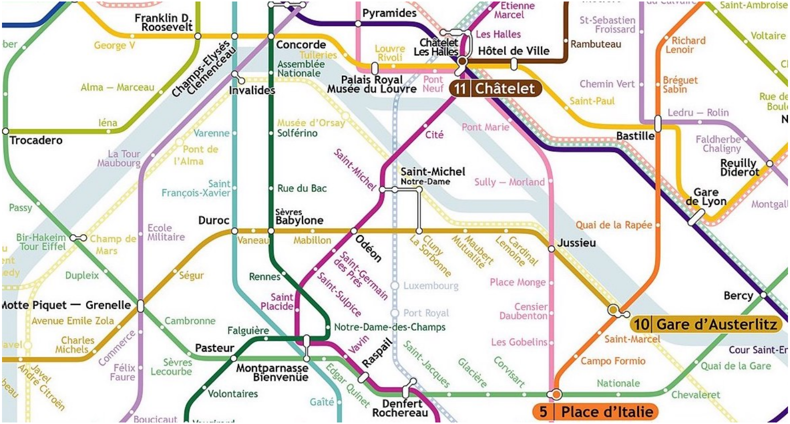
Output: Shortest path from s to everywhere

Notation

- $\to$ = edge
- $\rightsquigarrow$ = path
- $w(u, v)$ = weight of edge (u, v)
- $w(\overset{p}{\rightsquigarrow})$ = weight of path p = sum of weights of edges of p
- $v.d$ = "score" = Currently known weight of minimum-weight path from s to v.
- $\delta(s, v)$ = "min score" = shortest weight over all possible paths from s to v.

SSSP Example

What is the shortest path from Montparnasse to the Bastille?



From now on we assume edge weights are **strictly positive**.

Comparison to other SSSP

Name	BFS	Dijkstra	Bellman-Ford
Directed?	Both	Both	Both
Weights?	No	Yes	Yes
Negative weights?	N/A	No	Yes
Runtime:	$\Theta(n+m)$	$\Theta(n+m\log m)$	$\Theta(nm)$

Dijkstra vs. previous algorithms

Dijkstra vs. BFS:

- In **BFS** we first explore paths of increasing hop distance.
  When we discover a vertex we have its minimum hop path.
- In **Dijkstra** we explore paths of increasing **score**
  When we discover a vertex we have its minimum score.

Dijkstra vs. Bellman-Ford:

- In **Bellman-Ford** we simply relax all edges in arbitrary order
  Repeat until all scores are stable.
- In **Dijkstra** we relax edges outward from our source s
  Maintain a portion of SPT of s as we go.

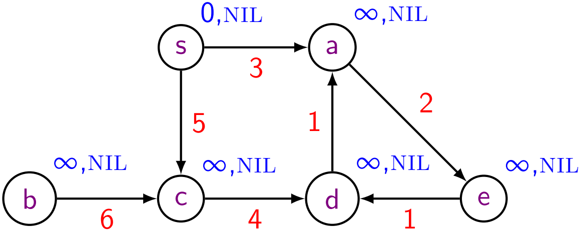
Dijkstra algorithm

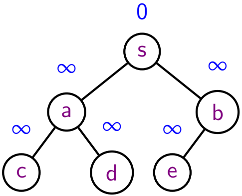
```
DIJKSTRA(G = (V, E), s, w)

Initialize s.d = 0, all other v.d = ∞. All parents NIL.
Q ← min-priority queue of all vertices (using score d as key).
S ← empty set S of processed elements.
While Q is not empty do:
    u ← EXTRACTMIN(Q)
    S ← S ∪ {u}
    For all edges (u, v) ∈ E:
        if (v.d > u.d + w(u, v))
            v.d ← u.d + w(u, v)
            v.π ← u
            FLOAT v in Q
```

Dijkstra - Example

Initial state. First we extract s.

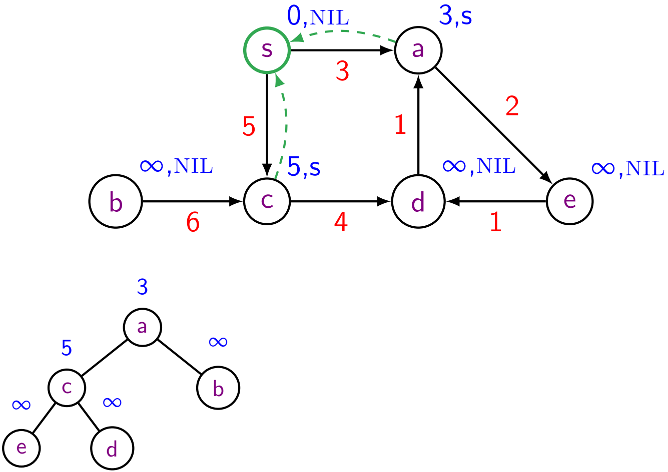




$S = \{\}$

Example

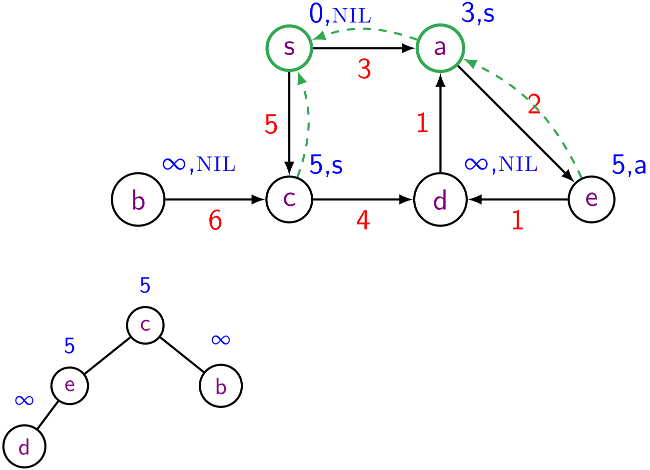
After extracting s (and updating adjacent vertices). Next is a.



$S = \{s\}$

Dijkstra - Example

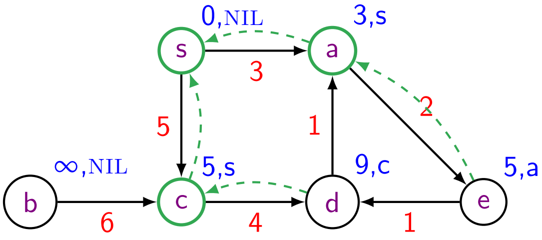
After extracting a (and updating adjacent vertices). Next is c.

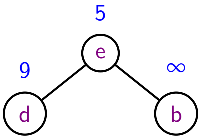


$S = \{s, a\}$

Dijkstra - Example

After extracting c (and updating adjacent vertices). Next is e.



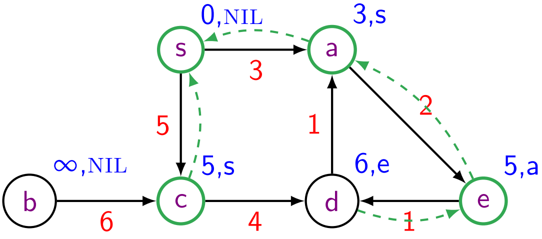


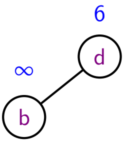
$S = \{s, a, c\}$

Notice every reachable node now has score and parent. Is this the final state?

Dijkstra - Example

After extracting e (and updating adjacent vertices). Next is d.

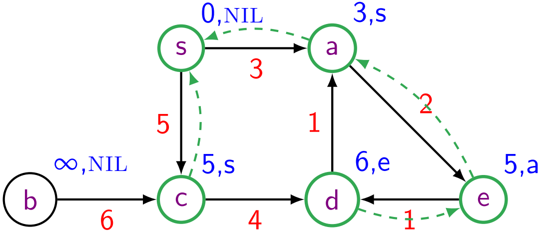


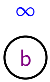


$S = \{s, a, c, e\}$ Aha!

Dijkstra - Example

After extracting d (and updating adjacent vertices). Next is b.

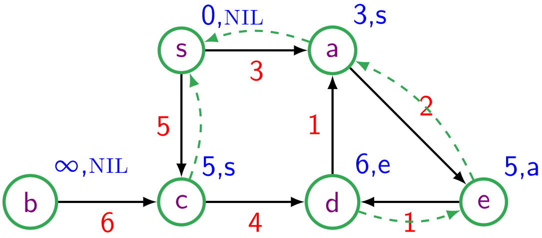




$S = \{s, a, c, e, d\}$

Dijkstra - Example

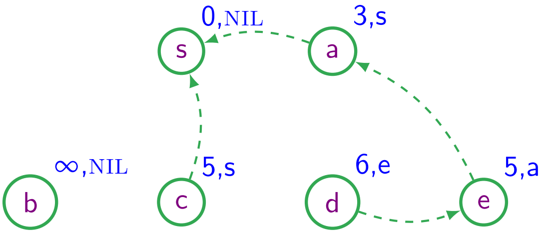
After extracting b (and updating adjacent vertices). Queue is empty.



$\varnothing$

$S = \{s, a, c, e, d, b\}$

Reporting solution



When Dijkstra ends, scores and parents have all information.

As with BFS/DFS/BF, we can transform to other formats.

Dijkstra - Correctness

Lemma

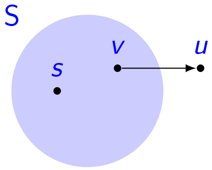
When we EXTRACTMIN *vertex u from Q we have $u.d = \delta(s, u)$. Thus u will have its correct min score when it is added to S.*

Proof: By induction as vertices are extracted from Q.

Base case: s is extracted first, and has its correct min score of 0.

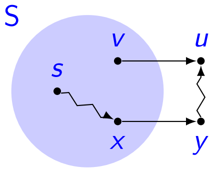
Inductive step:

- Inductive hypothesis: Assume all vertices added to S before u have the correct min score.
- u's current score comes from $u.\pi = v$ which is in S
- By IH, score of v is correct.
- **Claim**: any remaining path must cost more than $v.d + w(vu)$



Dijkstra - Correctness (cont.)

Claim: any remaining path must cost more than $v.d + w(vu)$



- Suppose there exists a shorter path $s \overset{p}{\rightsquigarrow} u$.
- p starts in S and ends at $u \notin S$
- $\Rightarrow$ there exists edge (x, y) in $s \overset{p}{\rightsquigarrow} u$ such that $x \in S$ and $y \notin S$.
- Consider two cases ($y = u$ and $y \neq u$):

$y = u$ then u should have its min score from x.
Contradiction to $u.\pi = v$.

$y \neq u$ then $y.d < u.d$.
Contradiction to $u \leftarrow \textsc{ExtractMin}(Q)$

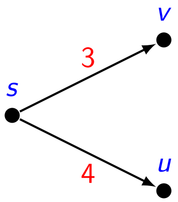
Runtime

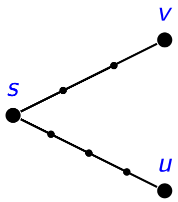
DIJKSTRA($G = (V, E), s, w$)

Initialize $s.d = 0$, all other $v.d = \infty$. All parents NIL.
$Q \leftarrow$ min-priority queue of all vertices (using score d as key).
$S \leftarrow$ empty set S of *processed* elements.
While Q is not empty do:
    $u \leftarrow$ EXTRACTMIN(Q)
    $S \leftarrow S \cup \{u\}$
    For all edges $(u, v) \in E$:
        if $(v.d > u.d + w(u, v))$
            $v.d \leftarrow u.d + w(u, v)$
            $v.\pi \leftarrow u$
            FLOAT v in Q

It's complicated. See **recitation**

Dijkstra Is Really BFS





- Dijkstra is an adaptation of BFS to weights

  Replace queue with priority queue

  Must RELAX to update info

  More powerful! Works for $w = \frac{1}{2}$ and $w = 10^{10}$

Discussion

Name	BFS	Dijkstra	Bellman-Ford
Directed?	Both	Both	Both
Weights?	No	Yes	Yes
Negative weights?	N/A	No	Yes
Runtime:	$\Theta(n+m)$	$\Theta(n+m\log m)$	$\Theta(nm)$

- BFS is greedy exploration
- Dijkstra is an adaptation of BFS to weights
- Bellman-Ford iteratively refines until no improvement possible
- Three different SSSP algorithms

  Faster algorithm $\Rightarrow$ more constrained
  Use the one that best suits the setting!

Additional practice questions

- Create a problem instance of ≈ 10 vertices.

  Run Dijkstra step by step on the instance
- What is the exact runtime of Dijkstra?
- Suppose $s \overset{p}{\leadsto} v$ is a shortest path from s to v. Does Dijkstra always relax the edges of p in order along the path? Why?
- Where in the correctness proof we use nonnegative weights?
- What happens if some edge weights are zero?
- Give a problem instance with negative weights in which Dijkstra fails
- Would Dijkstra work with undirected edges?
- Can Dijkstra handle graphs that are not connected?
- How would you report the shortest path tree after Dijkstra?