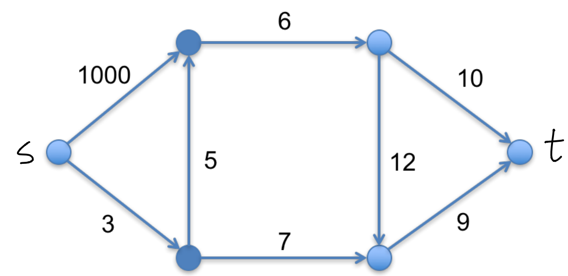
Goals

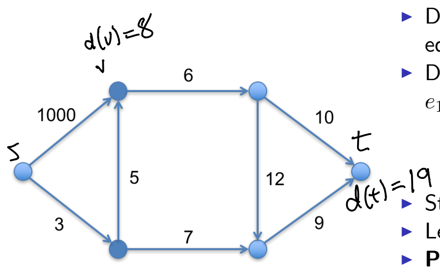
- Introduce the shortest path problem in directed graphs with nonnegative edge lengths

Shortest Paths Problem

Problem: find shortest paths in a directed graph with edge *lengths* (e.g., Google maps)



Let's Formalize the Problem



- Directed graph $G = (V, E)$ with **nonnegative** edge *lengths* $\ell(e) \geq 0$ $\ell(u,v)$ $e=(u,v)$
- Define *length* of path P consisting of edges $e_1, e_2, \ldots, e_k$ as

$$\ell(P) = \ell(e_1) + \ell(e_2) + \ldots + \ell(e_k)$$

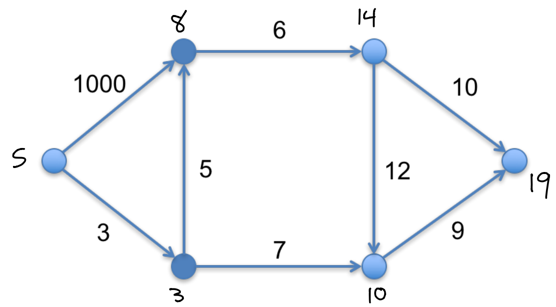
- Starting node s
- Let $d(v)$ be the length of shortest $s \rightsquigarrow v$ path.
- **Problem:** Can we efficiently find $d(v)$ for all nodes $v \in V$?
- **Question:** Why for all nodes at the same time?

Goals

- Derive Dijkstra's algorithm for shortest paths

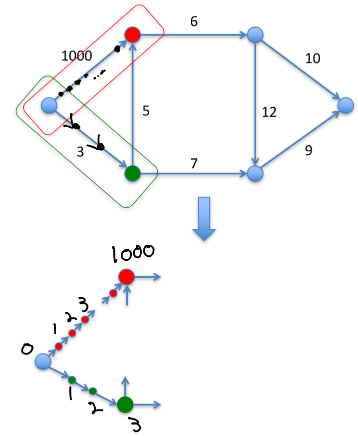
Dijkstra Derivation

Suppose all edges have integer length. Can we use BFS to solve this problem?

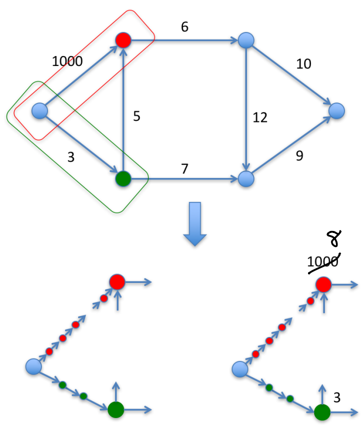


Recall: nodes in layer L_i are at distance i from start.

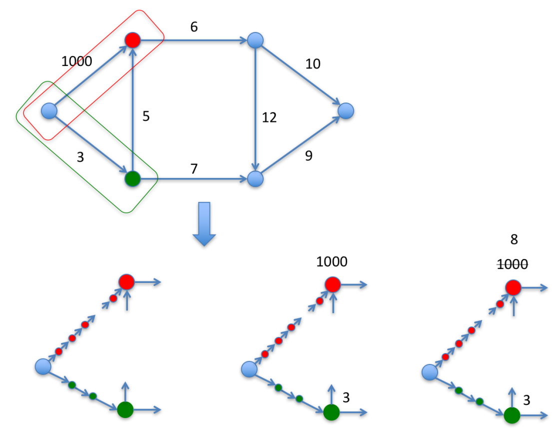
Dijkstra Derivation



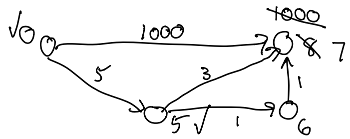
Dijkstra Derivation



Dijkstra Derivation



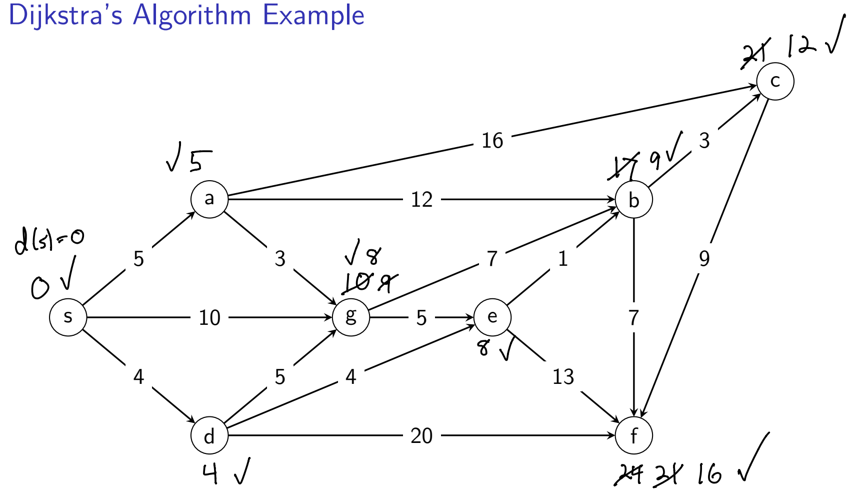
Dijkstra's Algorithm



Idea: keep track of expanding "wavefront" (arrival time at $v = d(v)$)

- find next node v to be hit by wave (= unexplored node closest to s)
- explore v: update tentative arrival time at each neighbor

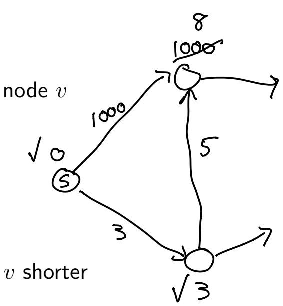
Dijkstra's Algorithm Example



Goals

- Describe implementation of Dijkstra's algorithm and analyze its running time

Dijkstra's Algorithm Implementation



Notation:

- $d'(v)$ — earliest tentative arrival time so far for node v
- $d(v)$ — shortest distance (actual arrival time)

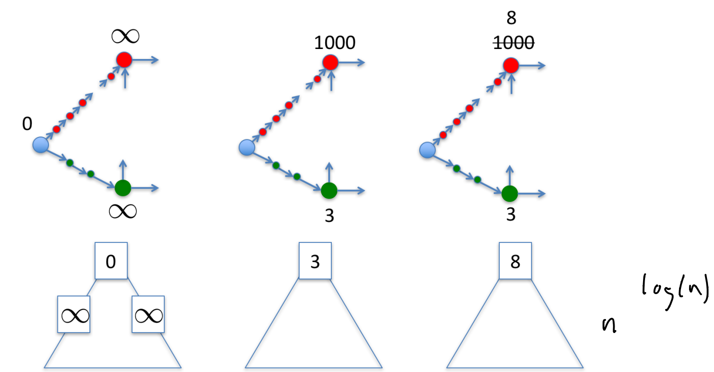
How to keep track of the wavefront?

- Find *next arrival*: node v with smallest $d'(v)$
- Set shortest distance: $d(v) = d'(v)$
- Update $d'(v)$ for neighbors of v if path through v shorter

What data structure supports find smallest and update values?

priority queue

Shortest Paths Problem



Dijkstra's Algorithm

unexplored nodes

set $A = V$ ▷ Priority queue
set $d'(v) = \infty$ for all nodes ▷ Tentative arrival time
set $d'(s) = 0$
while A not empty **do** ▷ Nodes left to explore
  extract node $v \in A$ with smallest $d'(v)$ value
  set $d(v) = d'(v)$ ▷ Wave arrives at v
  for all edges (v, w) where $w \in A$ **do**
    if $d(v) + \ell(v, w) < d'(w)$ **then** ▷ Shorter path to w??
      $d'(w) = d(v) + \ell(v, w)$

Running Time?

Use heap-based priority queue for A

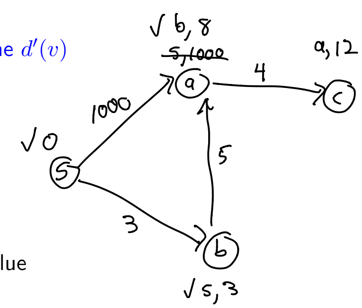
set $A = V$
set $d'(v) = \infty$ for all nodes
set $d'(s) = 0$
while A not empty **do**
  extract node $v \in A$ with smallest $d'(v)$ value n ▷ Extract-min
  set $d(v) = d'(v)$
  for all edges (v, w) where $w \in A$ **do** m
    if $d(v) + \ell(v, w) < d'(w)$ **then**
      $d'(w) = d(v) + \ell(v, w)$ $\leq m$ ▷ Update-key

- n extract-min operations. $O(n \log n)$
- m update-key operations. $O(m \log n)$
- **Total**: $O((m + n) \log n)$

Tracking the Shortest Path

Keep track of node that last updated arrival time $d'(v)$
Call it prev(v) = predecessor in shortest path

set $A = V$
set prev(v) = null for all nodes
set $d'(v) = \infty$ for all nodes
set $d'(s) = 0$
while A not empty **do**
  extract node $v \in A$ with smallest $d'(v)$ value
  set $d(v) = d'(v)$
  for all edges (v, w) where $w \in A$ **do**
    if $d(v) + \ell(v, w) < d'(w)$ **then**
      $d'(w) = d(v) + \ell(v, w)$
      prev$(w) = v$



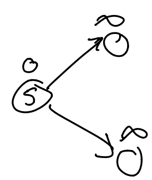
Goals

- Prove that Dijkstra's algorithm is correct

Proof of Correctness

- **Idea**: nodes are explored in increasing order of distance from s.
- At each step, we have:
  A: set of nodes still to explore
  $S = V \setminus A$: explored nodes, $d(v)$ assigned
- **Claim (invariant)**:
  (1) for $v \in S$, $d(v)$ is length of shortest $s \rightsquigarrow v$ path (correctness)
  (2) for $y \in A$, $d'(y)$ is the length of the shortest $s \rightsquigarrow y$ path with all nodes in S except y.
- Proof: By induction on $|S|$

Induction Proof



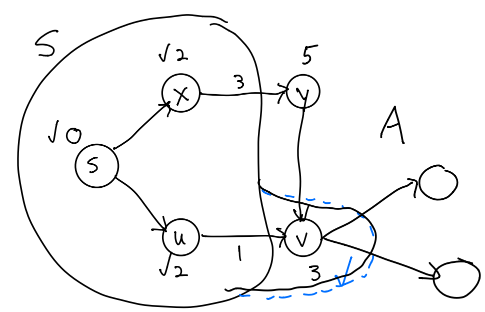
Base case: Initially $S = \emptyset$. Both (1) and (2) are true. ✓

Induction step:

- Assume the invariant is true for $|S| = k \geq 0$.
- Let v = next node added to S, so we set $d(v) = d'(v)$
- We claim (1) is true: the shortest $s \rightsquigarrow v$ path has length $d(v) = d'(v)$.
  - By (2), $d'(v)$ is length of the shortest path $s \rightsquigarrow v$ path with all prior nodes in S. We claim this is the shortest $s \rightsquigarrow v$ path overall.
  - Why? Every path to v must leave S eventually; if it first hops to some other node y outside of S, it is already at least as long as $d'(v)$, because $d'(v) \leq d'(y)$.
  - Paths can't get shorter after leaving S because edge lengths are non-negative.

1) $v \in S \Rightarrow d(v) =$ length of shortest $s \rightsquigarrow v$ path

2) $y \in A \Rightarrow d'(y) =$ length of shortest $s \rightsquigarrow y$ path with all nodes but $y \in S$



Induction Proof: Invariant (2)

Invariant (2) is directly maintained by the algorithm: after adding v to S, it updates $d'(w)$ for all neighbors w if a shorter path is found through v.

for all edges (v, w) where $w \in A$ **do**
  if $d(v) + \ell(v, w) < d'(w)$ **then**
    $d'(w) = d(v) + \ell(v, w)$