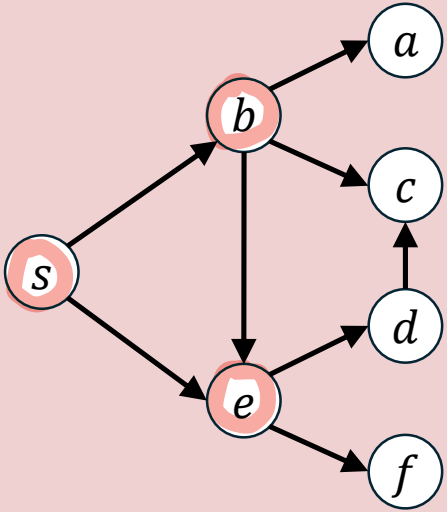
Poll Everywhere

PollEv.com/javabear text **javabear** to 22333

Which of the following *cannot* be the visitation order of a BFS of the following graph starting from source vertex s?



a c e b s

(A) s, b, e, a, c, d, f

(B) s, e, b, f, d, c, a

(C) s, b, e, c, d, a, f

(D) s, e, b, d, f, a, c



Lecture 23: Shortest Paths

CS 2110

November 12, 2026

Today's Learning Outcomes

80. Identify substructures in graphs such as neighborhoods, paths, and cycles both visually and programmatically.

83. Visualize the state of a graph traversal at a given point of its execution, describing each node as either undiscovered, discovered, visited, and/or settled.

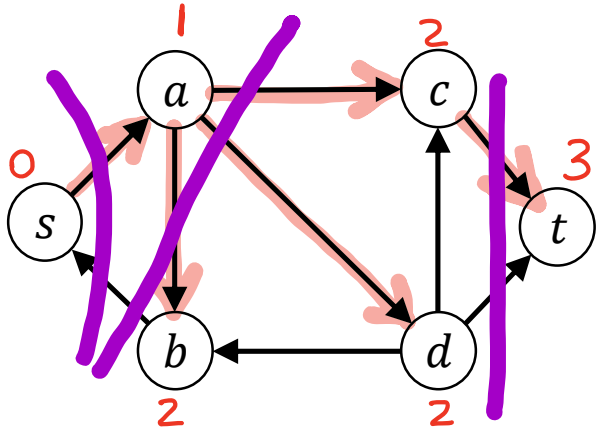
84. Describe Dijkstra's shortest path invariant and use it to establish properties on the unvisited portions of a graph at a given point in the execution of Dijkstra's algorithm.

85. Implement Dijkstra's shortest path algorithm and analyze its time/space complexities.

Traversal Levels

The **level** of a vertex v, $\ell(v)$, in a graph traversal starting at s is the length of the shortest $s \rightsquigarrow v$ path.

*BFS visits vertices in increasing level order.



Proof Idea: Think in phases

1. Visiting s (level 0) discovers all vertices at level 1, adding them to frontier
2. Visiting level 1 vertices discovers all level 2 vertices
   ⋮

Poll Everywhere

PollEv.com/javabear text **javabear** to 22333

Suppose (u, v) is an edge in a directed graph. What is the strongest statement we can make about the traversal levels of u and v, $\ell(u)$ and $\ell(v)$?

Imagine point when u was visited.

- If v undiscovered it will be in level $\ell(u)+1$
- If v discovered it was labeled earlier, so will be in level $\leq \ell(u)+1$

$\ell(v) = \ell(u) + 1$ **(A)**

$\ell(v) \geq \ell(u) + 1$ **(B)**

$\ell(v) \leq \ell(u) + 1$ **(C)**

$\ell(v) \geq \ell(u)$ **(D)**



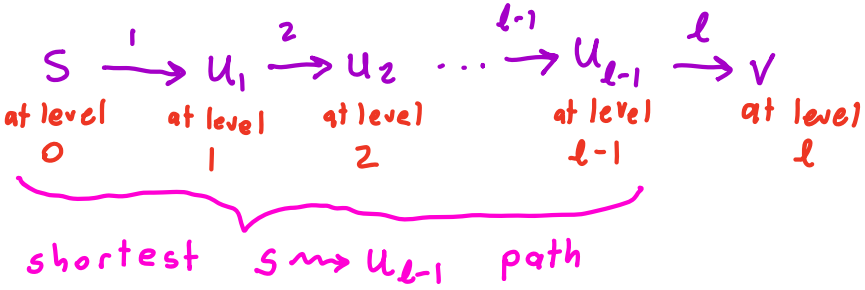
Coding Demo: Level-Augmented BFS

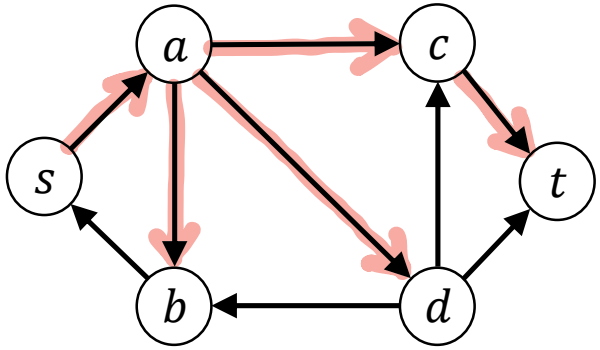


Key idea: Replace discovered Set with Map whose values keep track of levels

Shortest Paths in Unweighted Graphs

For a vertex v at level ℓ, the shortest $s \rightsquigarrow v$ path includes ℓ edges (by the definition of ℓ).





To recover all shortest paths from s, each vertex v should keep track of last edge in shortest $s \rightsquigarrow v$ path

For BFS, the edge that discovered the vertex works.



Coding Demo: BFS with PathInfo



Key ideas:

- PathInfo record class combines level + prev edge
- Update discovered Map to have PathInfo values

Reconstructing Shortest Paths

Exercise: Use a loop to compute `path` in this method definition.

```
record PathInfo<V>(int level, V prev) {}

/** Reconstructs and returns the shortest `src` -> `dst` path
 *  using the given `info` map produced by BFS. */
static <V> List<V> reconstructPath(Map<V,PathInfo<V>> info, V src, V dst) {
  List<V> path = new LinkedList<>();

  return path;
}
```

Map returned by BFS

Poll Everywhere

PollEv.com/javabear text **javabear** to 22333



What is the guard on your loop?

```
record PathInfo<V>(int level, V prev) {}

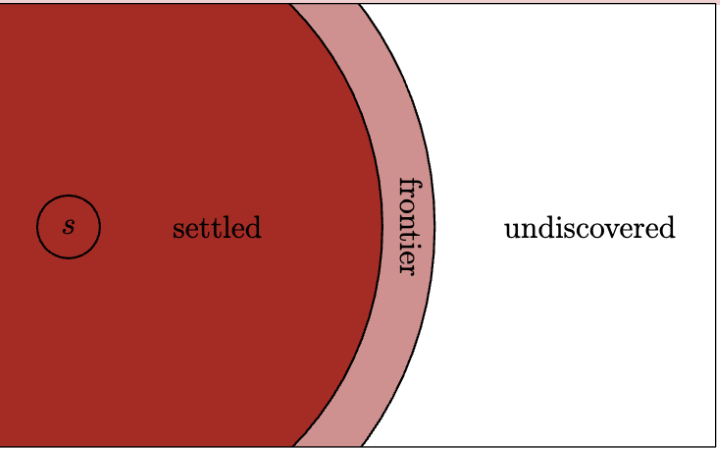
/** Reconstructs and returns the shortest `src` -> `dst` path
 *  using the given `info` map produced by BFS. */
static <V> List<V> reconstructPath(Map<V,PathInfo<V>> info, V src, V dst) {
  List<V> path = new LinkedList<>();
  path.add(dst);
  while (!path.getFirst().equals(src)) {
    path.addFirst(info.get(path.getFirst()).prev());
  }
  return path;
}
```

* build path backwards

first vertex

its (level, prev) info

Unweighted Shortest Paths (BFS) Invariants



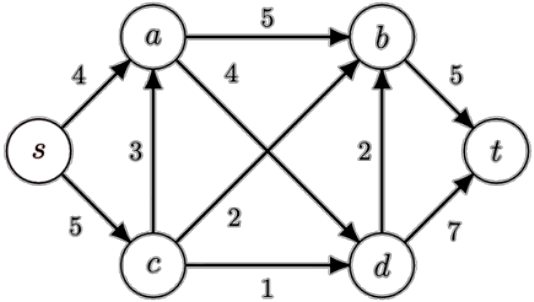
1. `discovered` map records shortest (known) path distance to every discovered vertex

2. Vertex v `dequeue()`d from `frontier` is *unvisited* vertex with min level

3. If a vertex v is settled, we've computed the shortest $s \rightsquigarrow v$ path, which contains only settled vertices

Adapting to Weighted Graphs (*non-negative weights)

High-Level Structure is the same as BFS

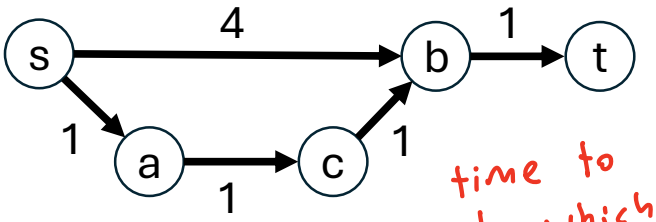


- Track `discovered` and `frontier` vertices
  - discovered Map will record shortest known s~>v distance instead of $\ell(v)$
- Visit/settle one `frontier` vertex in each main loop iteration
- During visit, use outgoing edges to discover new vertices and update `discovered` map and `frontier`

If edge (u,v) discovers v ⟹ v's entry in discovered Map has
dist = dist(u) + w(u,v) (edge length)
prev = u

Adapting to Weighted Graphs (*non-negative weights)

Issue 1: Distance order may differ from discovery order



discovered

vertex	s	a	b	c	t
dist	0	1	4	2	5
prev	null	s	s	a	b

no opportunity to find shorter $s \to a \to c \to b \to t$ path

Issue 2: May locate shorter path to vertex later in the traversal

need a way to update entries already in map

Poll Everywhere

PollEv.com/javabear text `javabear` to 22333



Which data structure should we use to model the `frontier` in Dijkstra's algorithm?

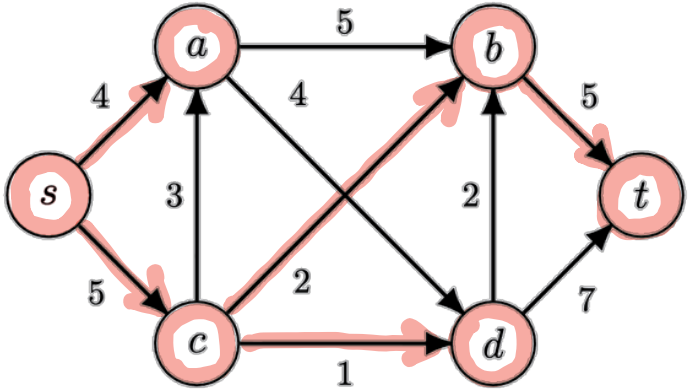
- Queue — visit order different from discovery order (A)
- Stack (B)
- Priority Queue — priorities change as we find new paths (C)
- Dynamic Priority Queue (D)



Coding Demo: Dijkstra's Algorithm



Dijkstra's Algorithm Walkthrough

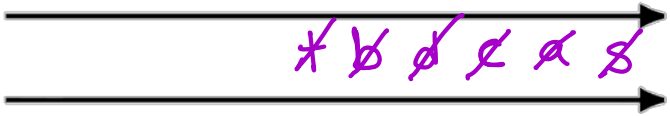


shortest path tree from s

discovered

vertex	s	a	b	c	d	t
distance	0	4	7	5	6	12
prev	null	s	c	s	c	b

frontier



* The lecture notes have an animation of this example.

Dijkstra's Invariant

At the start of each main loop iteration:

1. We've located the shortest path to all *settled* vertices.
2. The shortest *known* path to the *closest* `frontier` vertex v^* is guaranteed to be the true shortest $s \rightsquigarrow v^*$ path.

Proof Sketch:

- consider any alternate path to v^*
- at some point, this path must cross boundary to some frontier vertex v'

