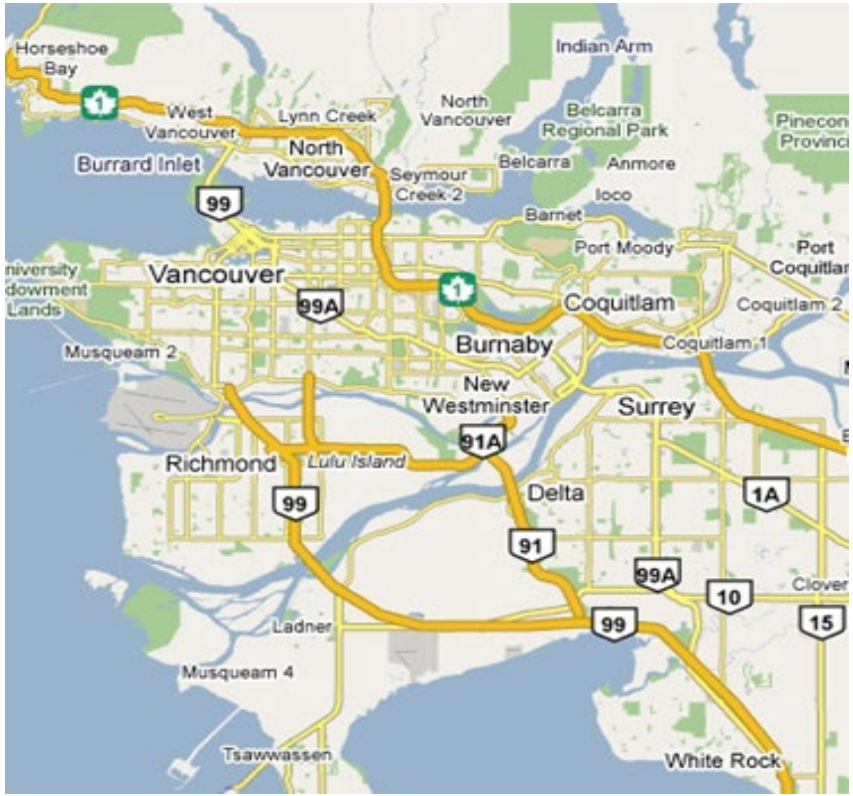
Programming, Problem Solving, and Algorithms

CPSC 203, 2024 W2
(January – April 2025)
Ian M. Mitchell
Lecture 12B

Slides from the Assigned Videos

Single Source Shortest Path

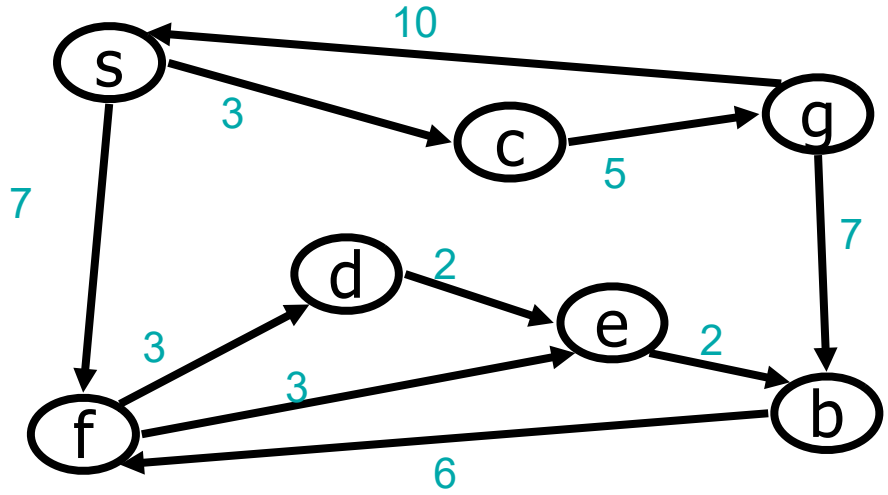


Given a start vertex (source) s, find the path of least total cost from s to every vertex in the graph.

Single Source Shortest Path

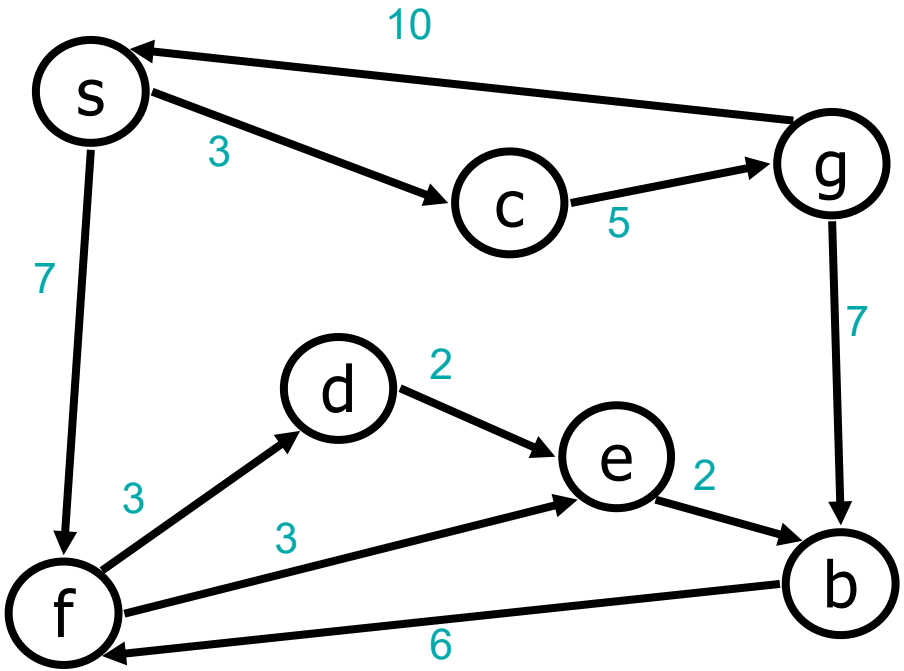
Input: directed graph G with non-negative edge weights, and a start vertex s.

Output: A subgraph G' consisting of the shortest (minimum total cost) paths from s to every other vertex in the graph.



Dijkstra's Algorithm (1959)

Single Source Shortest Path



Given a source vertex s, we wish to find the shortest path from s to every other vertex in the graph.

Initialize structure:

Repeat these steps:

1. Label a new (unlabelled) vertex v, whose shortest distance has been found
2. Update v's neighbors with an improved distance

Single Source Shortest Path

Initialize structure:

1. For all v, d[v] = “infinity”, p[v] = null
2. Initialize source: d[s] = 0
3. Initialize priority (min) queue

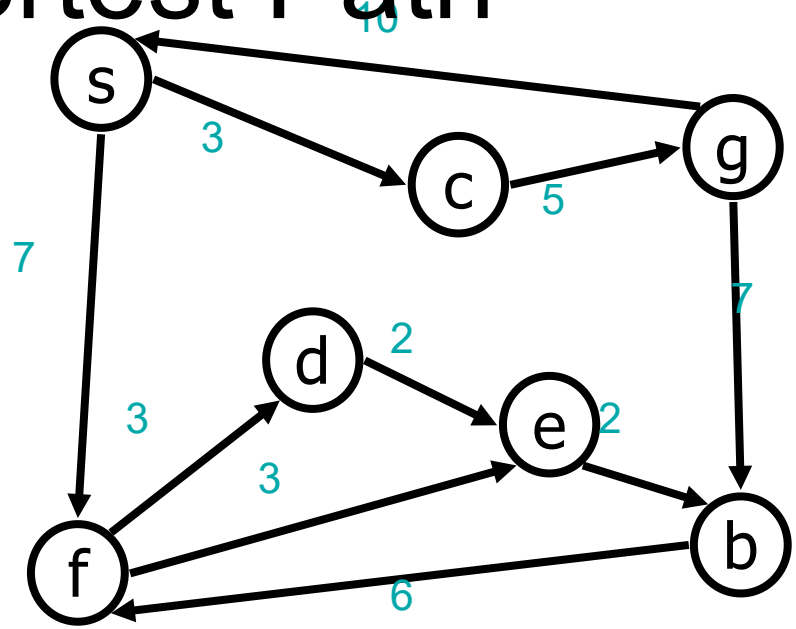
Repeat these steps n times:

- Find minimum d[] unlabelled vertex: v
- Label vertex v
- For all unlabelled neighbors w of v,

If (_____ < d[w])

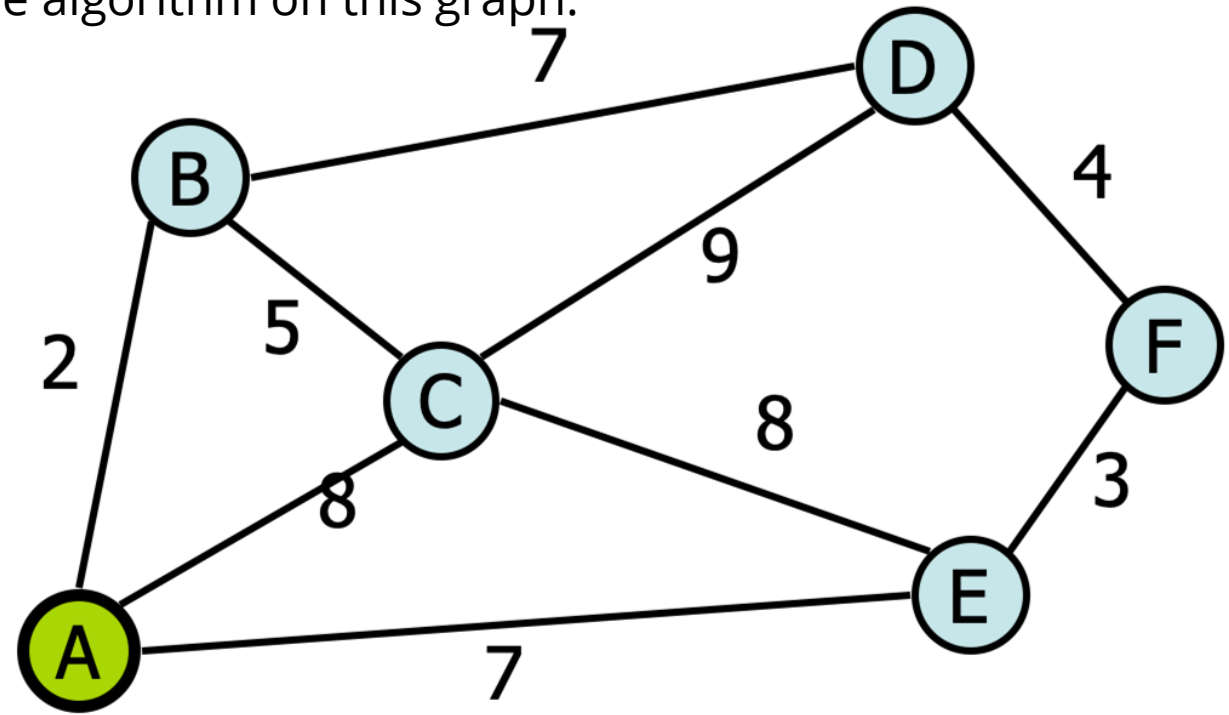
d[w] = _____

p[w] = v



Your Turn...

Execute the algorithm on this graph:



Dijkstra's Algorithm

How is this algorithm similar to BFS/DFS?

_____

_____

How is this algorithm different from BFS/DFS?

_____

_____

Initialize structure:

1. For all v, d[v] = "infinity", p[v] = null
2. Initialize source: d[s] = 0
3. Initialize priority (min) queue
4. Initialize set of labeled vertices to ∅.

Repeat these steps n times:

- Find & remove minimum d[] unlabelled vertex: v
- Label vertex v
- For all unlabelled neighbors w of v,

  If cost(v,w) < d[w]

  d[w] = cost(v,w)

  p[w] = v

Priority Queues

- Basic operations: push, pop, is-empty
  - Python (FIFO) queue using `deque q`: `q.appendleft, q.pop, q`
  - Python stack using `deque` (or `list`) `s`: `s.append, s.pop, s`
- What about priority queue?
  - Try a `list`:
  - Try a `deque`: