# Programming, Problem Solving, and Algorithms

CPSC203, 2019 W1

# Announcements

Project 3 released soon. Due 11:59p, Nov 29.

“Problem of the Day” continues!

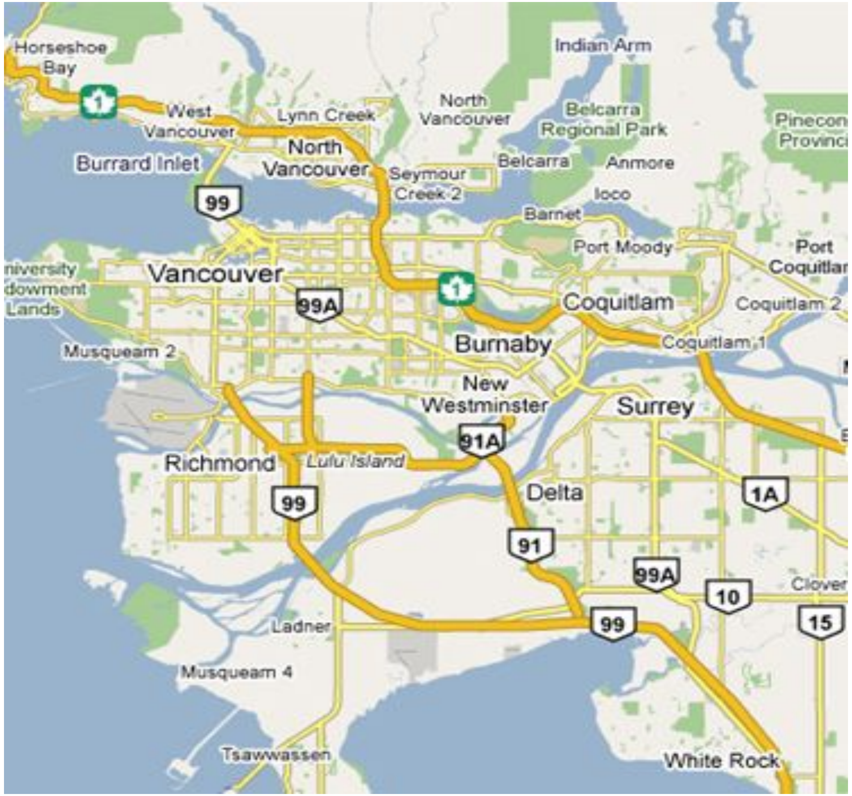
# Today:

Shortest Path

Maps!

How many Starbucks are in Vancouver?

# Single Source Shortest Path

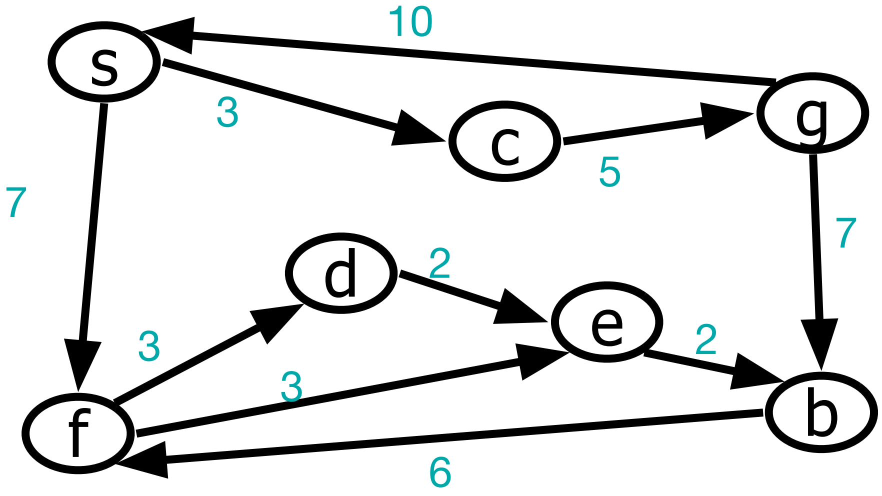



Given a start vertex (source) s, find the path of least total cost from s to every vertex in the graph.

# Single Source Shortest Path

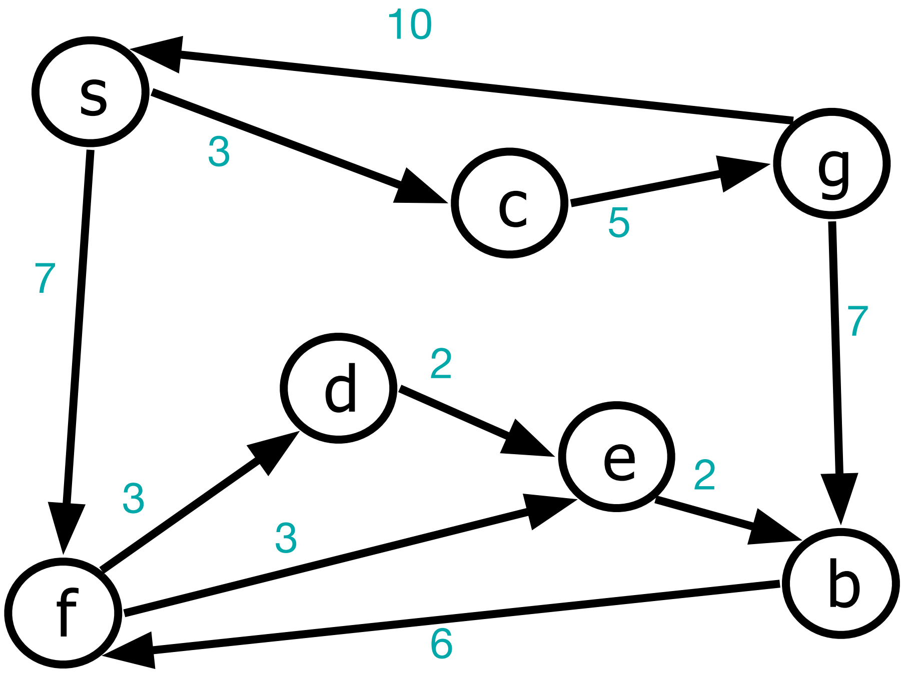
Input: directed graph G with non-negative edge weights, and a start vertex s.

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

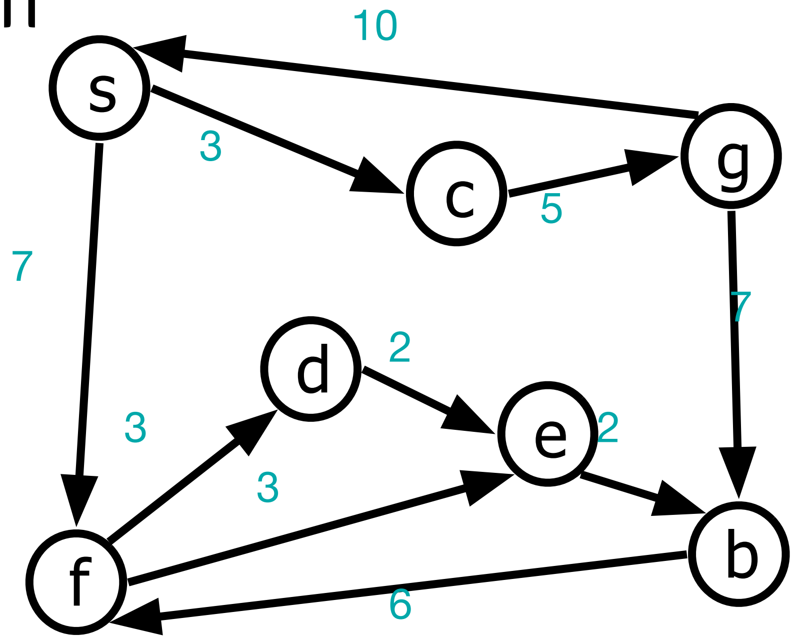
Repeat these steps n times:

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

  If (_________________ < d[w])

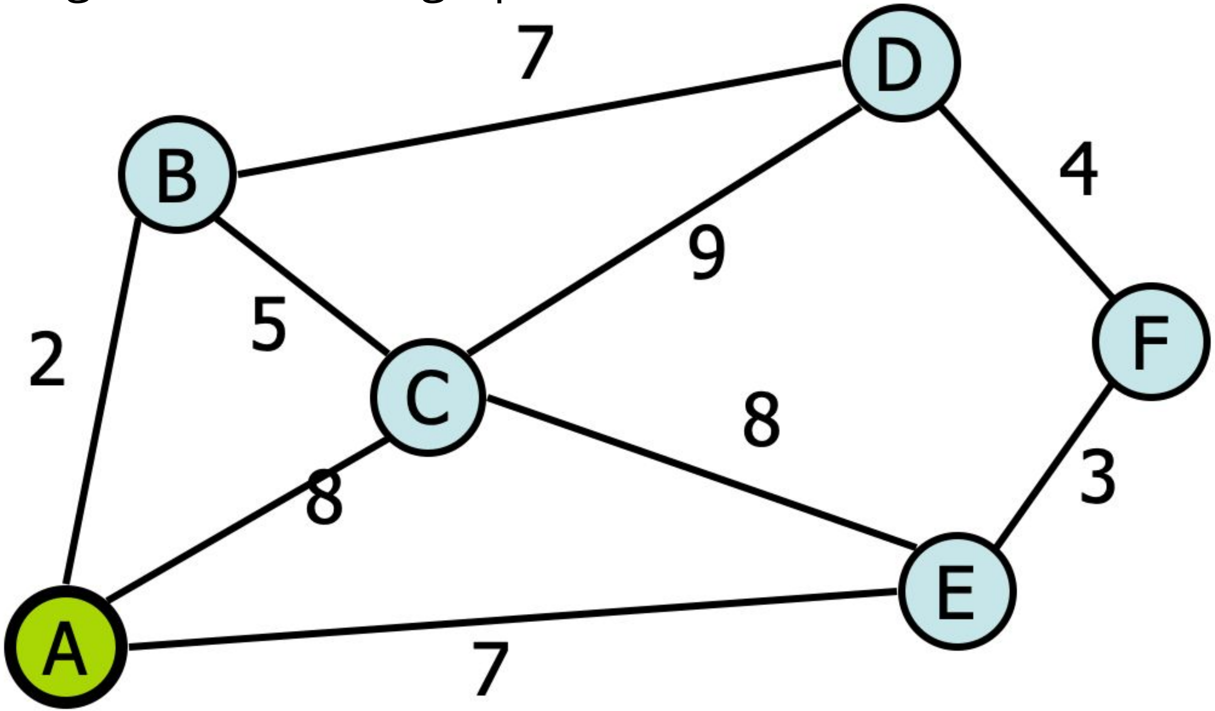
  d[w] = _______________

  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

# Map applications

## Three parts:

1. Assembling the data - OSM, local data stores, statsCan, etc. This is mostly the art of assembling geodataframes.

2. Computing on the data - osmnx simplifies graph algorithms and computation, but also supports other spatial computation.

3. Visualizing the data - matplotlib for static maps, folium for interactive maps.

# POTD #36 Thu

https://github.students.cs.ubc.ca/cpsc203-2019w-t1/potd36

Describe any snags you run into:

1. Line ___: ________________________________________
2. Line ___: ________________________________________
3. Line ___: ________________________________________
4. Line ___: ________________________________________
5. Line ___: ________________________________________

# ToDo for next class...

POTD: Continue every weekday! Submit to repo.

Reading: TLACS Ch 10 & 12 (lists and dictionaries)

References:

https://www.youtube.com/watch?v=wsSEKm-rU6U

https://github.com/gboeing/osmnx-examples/tree/master/notebooks

https://gist.github.com/psychemedia/b49c49da365666ba9199d2e27d002d07