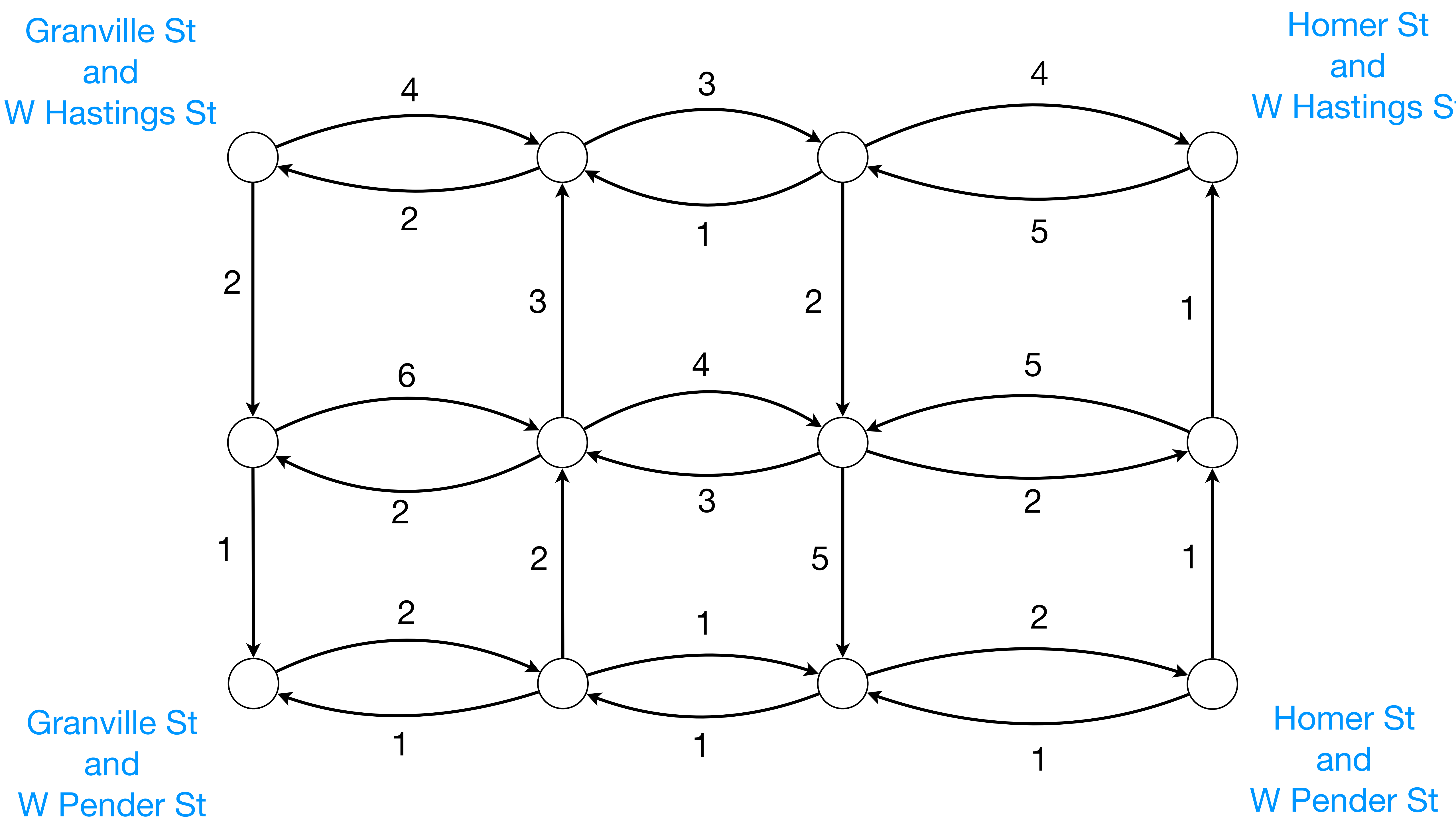
Shortest Paths

Nishant Mehta

Lecture 5 - Part II

Finding the Fastest Way to Travel between Two Intersections in Vancouver



Shortest Paths in Weighted Graphs

- Find fastest way to travel across the country using directed graph representing roads, with edge weights representing:
  - distances
  - travel times between cities
    (might account for speed limits, traffic, etc.)
- Find a fastest way using flights
  - Flight distances between airports.
  - Might also allow for warps in space-time continuum.
    Negative travel time

Single-Source Shortest Paths

- **If graph is unweighted:**
  - Breadth-First Search is a solution (more on this soon)
- **If graph is weighted:**
  - Every edge is associated with a number:
    integers, rational numbers, real numbers (might be negative!)
  - An edge weight can represent:
    distance, connection cost, affinity

Single-Source Shortest Paths Problem

- **Input:** A weighted directed graph G = (V, E)
  and a source vertex s

- **Output:** All single-source shortest paths for s in G, i.e., for all other vertices v in G, a shortest path from s to v.

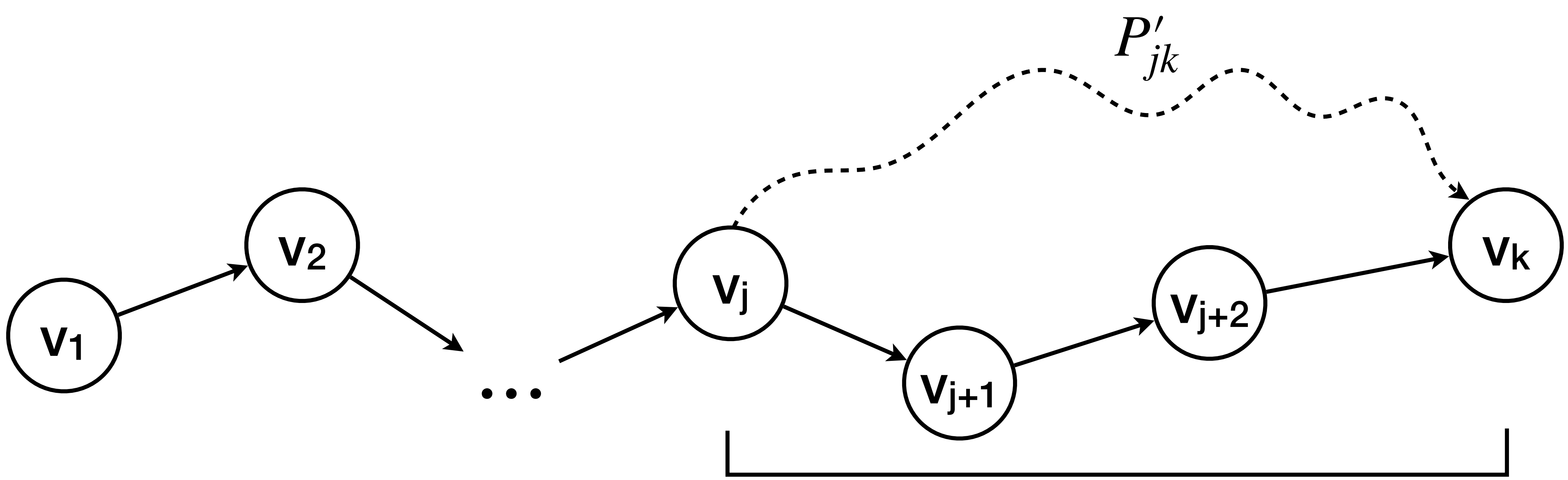
  - A path $p = (v_0, v_1, \ldots, v_k)$ from $s = v_0$ to $v = v_k$ is shortest if its length $w(p) = \sum_{j=1}^{k} w(v_{j-1}, v_j)$ is the minimum possible among all s–v paths

Optimal substructure

Optimal substructure - an optimal solution to a problem contains with it optimal solutions to subproblems

Example:

- Problem: Find shortest path from vertex v_1 to vertex v_k
- Subproblem: Find shortest path from intermediate vertex v_j to v_k



$$P_{jk} = (v_j,\ v_{j+1},\ v_{j+2},\ v_k)$$

Subpaths of shortest paths are shortest paths

Lemma

Let $P_{1k} = (v_1, v_2, \ldots, v_k)$ be a shortest path from v_1 to v_k.
Take some arbitrary i, j satisfying $1 \leq i < j \leq k$, and let
$P_{ij} = (v_i, v_{i+1}, \ldots, v_j)$ be the subpath of P_{1k} from v_i to v_j.
Then P_{ij} is a shortest path from v_i to v_j.

Relax: The most important function for today's lecture

RELAX(u, v)

If d[u] + w(u, v) < d[v]

$d[v] \leftarrow d[u] + w(u, v)$

$\pi[v] \leftarrow u$

Single-source shortest paths for weighted DAGs

- Suppose we have a weighted directed acyclic graph (DAG)
- An easy way to solve single-source shortest paths problem:

  (1) Use topological sort to obtain topological ordering (basically, use DFS + a slight amount of extra work)

  (2) For each vertex u in topological order

  For all vertices v adjacent to u

  RELAX(u, v)

- Runtime?

Single-source shortest paths for weighted DAGs

- Suppose we have a weighted directed acyclic graph (DAG)
- An easy way to solve single-source shortest paths problem:

  (1) Use topological sort to obtain topological ordering
  (basically, use DFS + a slight amount of extra work)

  (2) For each vertex u in topological order

  For all vertices v adjacent to u

  RELAX(u, v)

- Runtime? O(V + E)

Single-source shortest paths for weighted DAGs

- Suppose we have a weighted directed acyclic graph (DAG)
- An easy way to solve single-source shortest paths problem:

  (1) Use topological sort to obtain topological ordering (basically, use DFS + a slight amount of extra work)

  (2) For each vertex u in topological order

  For all vertices v adjacent to u

  RELAX(u, v)

- Runtime? O(V + E)
- Claim: Above algorithm is correct. Let's prove it!

Breadth-First Search for Unweighted Graphs

For each vertex, keep track of a color:

- White: Unvisited
- Red: Visited and Active - some adjacent vertices might not been added to queue yet
- Black: Visited and Inactive - all adjacent vertices have been added to queue

Pseudocode:

1. For all $u \in V$
    2. Color u White, set $d[u] = \infty$, and set $\pi[u] = \text{null}$
3. Color s Red and set $d[s] = 0$
4. Enqueue s into empty queue Q
5. While Q is not empty:
    6. $u \leftarrow \text{Dequeue}(Q)$
    7. For each White vertex v adjacent to u
        8. Color v Red
        9. Set $d[v] = d[u] + 1$ and $\pi[v] = u$.
        10. Enqueue v into Q
    11. Color u Black