Lecture 9: Overview

- Single Source Shortest Path
Problem

• Single starting vertex
• Find shortest path to all other vertices
Initialization

- All algorithms in chapter use same basic strategy: relaxation

\[ \text{INITIALIZE-SINGLE-SOURCE}(G, s) \]

\[
\text{for each vertex } v \in V[G] \\\n\quad \text{do } d[v] <- \infty \\
\qquad \text{pr}[v] <- \text{NIL} \\
\text{d}[s] <- 0
\]
RELAX(u, v, w)
1 if d[v] > d[u] + w(u, v)
2 then d[v] <- d[u] + w(u, v)
3 pr[v] <- u
Bellman-Ford Algorithm

BELLMAN-FORD(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 for i<- 1 to |V[G]| -1
3   do for each edge (u,v) ∈ E[G]
4      do RELAX(u, v, w)
5   for each edge (u,v) ∈ E[G]
6      do if d[v] > d[u] + w(u,v)
7         then return FALSE
8 return TRUE
Bellman-Ford Algorithm

- Works for edges with negative lengths
- Problem isn’t well-defined if there is a cycle of negative length
- If there is a negative cycle, algorithm returns false
- Complexity is $O(VE)$
Shortest Path in DAG

- If we consider nodes of a weighted DAG in topologically sorted order, we can compute shortest path in $\Theta(V+E)$ time

```
DAG-SHORTEST-PATHS(G, w, s)
1 topologically sort the vertices of G
2 INITIALIZE-SINGLE-SOURCE(G,s)
3 for each vertex u, taken in topologically sorted order
4   do for each vertex v ∈ Adj[u]
5     do RELAX(u, v, w)
```
Dijkstra’s Algorithm

DIJKSTRA(G, w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S <- {}
3 Q <- V[G]
4 while Q ≠ {}
5 do u <- EXTRACT-MIN(Q)
6 S <- S ∪ {u}
7 for each vertex v ∈ Adj[u]
8 do RELAX(u, v, w)
Dijkstra’s Algorithm

- Idea is to keep distance estimates to neighbors of set S
- Add neighbor with shortest distance to set S and update other neighbors
- Doesn’t work with negative edges (they could affect distances in S)
- Exact runtime depends on implementation of priority queue
Difference Constraints and Shortest Paths

- Want to satisfy a set of constraints \( Ax \leq b \)
- Special case - difference constraints
- Each row of A contains one 1 and one -1
- Can set up as a shortest paths problem
- Variables are vertexes (plus special vertex \( v_0 \) which has zero weighted edges to all other vertices)
- Edges represent constraints
  - Constraint \( x_j - x_i \leq b_k \) is represented as an edge \((v_i, v_j)\) with weight \( b_k \) (correspond to triangle constraints)
- Use Bellman-Ford algorithm to satisfy