Shortest paths

**The Bellman-Ford algorithm**

- Most basic algorithm for the shortest-path problem
- Allow negative-weight edges
- Compute \(d[v]\) and \(\pi[v]\) for all \(v \in V\)
  - \(d[v] = \delta(s, v)\): the shortest-path weight from the source \(s\) to \(v\).
  - \(\pi[v]\): the parent (predecessor) of \(v\).
- Return **TRUE** if no negative-weight cycles reachable from source \(s\), **FALSE** otherwise.
Shortest paths

Bellman-Ford(G, w, s)
for each vertex v in V // initialization
    d[v] = infty
    pi[v] = nil
endfor

d[s] = 0
for i = 1 to |V|-1 // |V|-1 passes
    for each edge (u,v) in E // in a prescribed order
        if d[v] > d[u] + w(u,v) // relax if necessary
            d[v] = d[u] + w(u,v)
            pi[v] = u
        endiffor
    endfor
endfor
for each edge (u,v) in E // final check pass
    if d[v] > d[u] + w(u,v)
        return FALSE
    endiffor
return TRUE, d, pi
Shortest paths

The Bellman-Ford algorithm

- Run and illustrate the Bellman-Ford algorithm

- Running time: $\Theta(|V| \cdot |E|)$.

- Values you get on each pass and how quickly it converges depends on order of relaxation (processing edges). But guaranteed to converge after $|V| - 1$ passes, assuming no negative-weight cycles.
Shortest paths

Dijkstra’s algorithm

- No negative weight edges
- Like BFS. If all weights = 1, use BFS.
- Use $Q = \text{priority queue keyed by } d[v]$ (vs. BFS uses FIFO queue)
Shortest paths

Dijkstra(G, w, s)
for each vertex v in V // Initialization
    d[v] = infty
    pi[v] = nil
endfor
d[s] = 0
Q = V // priority queue keyed by d[v]
while Q is not empty
    u = Extract-Min(Q)
    for all edge (u,v) in E
        if d[v] > d[u] + w(u,v) // Relax if necessary
            d[v] = d[u] + w(u,v)
            pi[v] = u
        endif
    endfor
endwhile
return d, pi
Shortest paths

Dijkstra’s algorithm

- Run and illustrate Dijkstra’s algorithm

- Running time: $O(|E| \lg |V|)$ (binary heap)

- Similar to the BFS and MST-algorithms, Dijkstra’s algorithm is a greedy algorithm. It always chooses the “lightest” or “closest” vertex in $V - S$ to insert into $S$, where $S$ is the set of vertices whose final shortest-path weights are determined.
Shortest paths

The SSSP in DAG

- DAG: can have negative-weight edges, but no negative-weight cycle.

- How fast can do it?

  Answer: $O(|V| + |E|)$, instead of $\Theta(|V| \cdot |E|)$ by Bellman-Ford
Shortest paths

DAG-Shortest-Path(G, w, s)
Topological sort of the vertices of G
for each vertex v in V
    d[v] = infty
    pi[v] = nil
endfor

d[s] = 0
for each vertex u taken in topologically sorted order
    for each vertex v in Adj[u]
        if d[v] > d[u] + w(u,v)
            d[v] = d[u] + w(u,v)
            pi[v] = u
        endif
    endfor
endfor

return d, pi