CSCE 411
Design and Analysis of Algorithms

Set 5a: Bellman-Ford SSSP Algorithm
Slides by Prof. Jennifer Welch
Spring 2014
Bellman-Ford Idea

- Consider each edge (u,v) and see if u offers v a cheaper path from s
  - compare d[v] to d[u] + w(u,v)
- Repeat this process |V| - 1 times to ensure that accurate information propagates from s, no matter what order the edges are considered in
Bellman-Ford SSSP Algorithm

- input: directed or undirected graph G = (V,E,w)

// initialization
- initialize d[v] to infinity and parent[v] to nil for all v in V other than the source
- initialize d[s] to 0 and parent[s] to s

// main body
- for i := 1 to |V| - 1 do
  - for each (u,v) in E do // consider in arbitrary order
  - if d[u] + w(u,v) < d[v] then
    - d[v] := d[u] + w(u,v)
    - parent[v] := u
Bellman-Ford SSSP Algorithm

// check for negative weight cycles
for each (u,v) in E do
  if d[u] + w(u,v) < d[v] then
    output "negative weight cycle exists"
Running Time of Bellman-Ford

- O(V) iterations of outer for loop
- O(E) iterations of inner for loop
- O(VE) time total
Bellman-Ford Example

process edges in order
(c,b)
(a,b)
(c,a)
(s,a)
(s,c)

<board work>
Correctness of Bellman-Ford

Assume no negative-weight cycles.

**Lemma:** $d[v]$ is never an underestimate of the actual shortest path distance from $s$ to $v$.

**Lemma:** If there is a shortest $s$-to-$v$ path containing at most $i$ edges, then after iteration $i$ of the outer for loop, $d[v]$ is at most the actual shortest path distance from $s$ to $v$.

**Theorem:** Bellman-Ford is correct.

**Proof:** Follows from these 2 lemmas and fact that every shortest path has at most $|V| - 1$ edges.
Correctness of Bellman-Ford

- Suppose there is a negative weight cycle.
- Then the distance will decrease even after iteration $|V| - 1$
  - shortest path distance is negative infinity
- This is what the last part of the code checks for.
The previous example would have converged faster if we had considered the edges in a different order in the for loop
- move outward from s

If the graph is a DAG (no cycles), we can fully exploit this idea to speed up the running time
DAG Shortest Path Algorithm

- input: directed graph $G = (V,E,w)$ and source vertex $s$ in $V$
- topologically sort $G$
- $d[v] := \infty$ for all $v$ in $V$
- $d[s] := 0$
- for each $u$ in $V$ in topological sort order do
  - for each neighbor $v$ of $u$ do
    - $d[v] := \min\{d[v], d[u] + w(u,v)\}$
DAG Shortest Path Algorithm

- Running Time is $O(V + E)$.
- Example: <board work>