Bellman-Ford

INITIALIZE(V, E, s)

```
1  for v ∈ V
2      d[v] ← ∞          distance
3      π[v] ← NULL       parent pointer
4  d[s] ← 0
5  π[s] ← s
```

RELAX(u, v)

```
1  if d[v] > d[u] + w(u, v)
2      d[v] ← d[u] + w(u, v)
3      π[v] ← u
```

BELLMAN-FORD(V, E, s)

```
1  INITIALIZE(V, E, s)
2  for i = 1 : |V| − 1
3      for each edge (u, v) ∈ E
4          RELAX(u, v)
5  for each edge (u, v) ∈ E          } cycle detection
6      if d[v] > d[u] + w(u, v)
7          return FALSE
8  return TRUE
```

DAG-BELLMAN-FORD(V, E, s)

```
1  TOPOLOGICALLYSORT(V, E)
2  INITIALIZE(V, E)
3  for each vertex u ∈ V taken in topological order
4      for each edge originating at u, (u, v) ∈ E
5          RELAX(u, v)
```

Contents

1 Generic Algorithms

1.1 Data Structures and Notation

distance
parent
dist from s→v

- $d[v]$ is length of current shortest path from s to v (evolves over search) ✳
- $\pi[v]$ is current parent of v
- $\delta(s, v)$ is length of actual shortest path from s to v ✳

✳ Keys to correctness

1.2 Algorithm

For single-source shortest path when edges have negative weights (also works when they don't, but there are faster algorithms).

NOTE: If graphs have *negative weight cycles*, the shortest path isn't well defined, because you'd rather live in the cycle!

GENERIC-SHORTESTPATH(V, E, s)
1 INITIALIZE(V, E, s)
2 **while** there is an edge $(u, v) \in E$ s.t. $d[v] > d[u] + w(u, v)$
3 "somehow" select one such edge (u, v)
4 RELAX(u, v)

This never steps if you have negative weight cycles!

All shortest path algorithms are about specifying the "somehow".

A bad choice can give an exponential running time... That's very bad.

1.3 Theorems

Theorem 0: Running Time The recursion

$$T(n) = C_1 + C_2 T(n - C_3) \tag{1}$$

gives a running time exponential in n for $C_2 > 1$.

Proof: Look at the tree:

The first level has C_2 nodes. The second level has C_2^2 nodes. There are $n - C_3$ levels $= C_2^{n-C_3}$ work at the bottom level! FAIL.

intermediate shortest paths are themselves shortest paths

Theorem 1: Subpath Theorem Let $\{v_1, v_2, ..., v_n\}$ be a shortest path from v_1 to v_n. Then any subsequence of this path from v_i to v_j is a shortest path from v_i to v_j.

Proof: Assume the contrary. Assume there is a shortest path from v_i to v_j with length $l < L$ where L is the length along the path from v_i to v_j. Then we could create a shorter path from v_1 to v_n by inserting the shorter path from v_i to v_j.

Sum of shortest paths $(u,x)(x,v) \geq sp(u,v)$

Theorem: Triangle Inequality $\forall u, v, x \in V$, we have $\delta(u, v) \leq \delta(u, x) + \delta(x, v)$.

Proof: Assume the contrary. Let $\delta(u, v) > \delta(u, x) + \delta(x, v)$. Then the shortest path from u to x plus the shortest path from x to v has a shorter length than $\delta(u, v)$ contradicting that $\delta(u, v)$ is the length of the shortest path from u to v.

2 Bellman-Ford

2.1 Algorithm

See the first page.

Running time: Initialization: $O(|V|)$, Loop on line 3 $O(|V||E|)$, Negative cycle checking: $O(|E|) \Rightarrow O(|V||E|)$ running time

2.2 Correctness

at termination $d[v] = \delta(s,v)$

Lemma 1: Upper Bound Property We always have $d[v] \geq \delta(s, v)$ and if we ever find $d[v] = \delta(s, v)$, $d[v]$ never changes.

Proof: We show that $d[v] \geq \delta(s, s)$ by induction on the number of relaxation steps k.

Base Case: For $k = 0$, we have $d[s] = 0$ and $d[v] = \infty$ for all $v \in \{V \backslash \}$. If there is no negative cycle, then $\delta(s, s) = 0$; otherwise $\delta(s, s) = -\infty$ so $d[s] \geq \delta(s, s)$.

Induction Step: Assume after $k-1$ relaxations, $d[v] \geq \delta(s,v)$ for all $v \in V$. Now consider the kth relaxation on edge (u,v). Firstly, for $w \in \{V \setminus v\}$, $d[w]$ does not change so by induction $d[w] \geq \delta(s,w)$. Now consider we set $d[v] = d[u] + (u,v) \geq \delta(s,u) + (u,v) \geq \delta(s,v)$ where the last step follows by the triangle inequality.

taking edge from u→v for s→v path

relaxing doesnt increase d[v] & no neg-cycles => shortest dist cant get shorter

If we ever achieve $d[v] = \delta(s,v)$, we cannot decrease $d[v]$ since we have just shown that $d[v] \geq \delta(s,v)$. Moreover, we cannot increase $d[v]$ because we only change $d[v]$ during relaxations and relaxations cannot increase $d[v]$. Therefore, once we have $d[v] = \delta(s,v)$ it cannot change.

once it is relaxed, a path cannot get better

Lemma 2: Path Relaxation Assume we have a graph G with no negative cycles. Let $p = \langle v_0, v_1, ..., v_j \rangle$ be a shortest path from v_0 to v_j. Any sequence of calls to RELAX that includes, in order, the relaxations of $(v_0,v_1),(v_1,v_2),...,(v_{j-1},v_j)$ produces $d[v_j] = \delta(v_0,v_j)$ after all of these relaxations and at all times afterwards. Note that this property holds regardless of what other relaxation calls are made before, during, or after these relaxations.

Proof: We proceed by induction on the kth vertex in p, showing that after relaxations $(v_0,v_1),...,(v_{k-1},v_k)$ have occurred that $d[v_k] = \delta(v_0,v_k)$.

Base Case: $k = 0$: After initialization we have that $d[v_0] = 0 = \delta(v_0,v_0)$.

Induction Step: Assume that we have relaxed, in order, edges $(v_0,v_1),...,(v_{k-2},v_{k-1})$ and $d[v_{k-1}] = \delta(v_0,v_{k-1})$. Then eventually we will make a relaxation call to (v_{k-1},v_k) since we assume this call happens at least once after the call to (v_{k-2},v_{k-1}). By the upper bound property, at the time of this call $d[v_k] \geq \delta(v_0,v_k)$. In addition, we have that $\delta(v_0,v_k) = \delta(v_0,v_{k-1}) + w(v_{k-1},v_k)$ because p is a shortest path. Therefore, $d[v_k] \geq d[v_{k-1}] + w(v_{k-1},v_k)$ and after this relaxation call we will have $d[v_k] = d[v_{k-1}] + w(v_{k-1},v_k) = \delta(v_0,v_k)$. By the upper bound property, this is maintained ever after.

Previous $\delta(u,v) \Rightarrow$ new $\delta(u,v)$

$d[v] \geq \delta(u,v)$

Lemma 3: BELLMAN-FORD Correctness 1: Assume we have a graph G with no negative cycles reachable by s. Then after running BELLMAN-FORD, we have $d[v] = \delta(s,v)$ for all $v \in V$ reachable from s.

Proof: If $v \in V$ is reachable from s then there must exist an acyclic shortest path $p = \langle s, v_1, ..., v_j \rangle$ where $v_j = v$. Now, since p is acyclic, p can contain no more than V vertices and therefore no more than $|V|-1$ edges. Each time we run through the loop on line 3 of BELLMAN-FORD, we relax every edge. Therefore, we relax (s,v_1) on the first iteration, (v_1,v_2) on the second iteration, etc (we also relax (v_1,v_2) on the first iteration, of course, but we cannot guarantee that relaxation comes after the relaxation of (s,v_1)). By the time we have reached the $|V|-1$ iteration, we must have relaxed every edge in p in order. Therefore, by the path relaxation property, we have $d[v] = \delta(s,v)$ when we have done $|V|-1$ iterations of the loop on line 3.

Corollary 1: BELLMAN-FORD Correctness 2: For vertex $v \in V$, BELLMAN-FORD terminates with $d[v] = \infty$ if and only if v is not reachable from s.

Proof: Let $d[v] = \infty$ and assume v is reachable from s. Then Lemma 3 would give that $d[v] = \delta(s,v) = \infty$, which is a contradiction.

Assume v is not reachable from s. By the upper bound property, we have that $d[v] \geq \delta(s,v)$ at all times. Since $\delta(s,v) = \infty$ we must have $d[v] = \infty$ at all times.

$\delta(s,v) = -\infty$ for all $v \in V$

Theorem 1: Correctness of BELLMAN-FORD If G contains no negative cycles reachable from s, the algorithm returns TRUE and $d[v] = \delta(s, v)$ for all $v \in V$. If G does contain a negative weight cycle, the algorithm returns FALSE.

Proof: If G contains no negative cycles, by Lemma 3 and Corollary 1, $d[v] = \delta(s, v)$ for all $v \in V$ at the termination of the loop beginning in line 3. Therefore $d[v] = \delta(s, v) \leq \delta(s, u) + w(u, v) \leq d[u] + w(u, v)$ by the triangle inequality and the check on line 6 always fails.

Now assume G contains a negative weight cycle that is reachable from s. Let this cycle be $c = \langle v_0, v_1, ..., v_j \rangle$ where $v_0 = v_j$. Then, by definition, we must have $\sum_{i=1}^{j} w(v_{i-1}, v_i) < 0$. We proceed by contradiction. Assume BELLMAN-FORD returns TRUE. Then for all $v \in V$, the inequality on line 6 must hold. Specifically, we must have

$$\sum_{i=1}^{j} w(v_{i-1}, v_i) \leq \sum_{i=1}^{j} (d[v_{i-1}] + w(v_{i-1}, v_i)) \quad (2)$$

$$\leq \sum_{i=1}^{j} d[v_{i-1}] + \sum_{i=1}^{j} w(v_{i-1}, v_i) \quad (3)$$

since $v_0 = v_j$, $\sum_{i=1}^{j} d[v_i] = \sum_{i=1}^{j} d[v_{i-1}]$. Therefore, we find that $\sum_{i=1}^{j} w(v_{i-1}, v_i) \geq 0$, contradicting our original assumption that c was a negative cycle. Hence G cannot return TRUE and must return FALSE.

3 Bellman-Ford on a DAG

Wouldn't it be nice if we could ensure that we did things in the right order to fulfill the path relaxation property for all vertices by only looking at each edge *once*?

Yes.

Say we have a DAG. In what order should we look at the edges? From start to each vertex... Topological Sort!

See first page for algorithm.

Running Time: Topological sort: $O(|V| + |E|)$, Initialization: $O(|V|)$, Main algorithm: $O(|E|) \Rightarrow O(|V| + |E|)$ Linear time!!

Correctness: At the terminiation of DAG-BELLMAN-FORD, $d[v] = \delta(s, v)$ for all $v \in V$.

Proof: If v is not reachable from s then $d[v] = \infty$ since $d[v] \geq \delta(s, v)$ by the upper bound property. Now assume v is reachable from s and that a shortest path from s to v is $\langle v_0, v_1, ..., v_j \rangle$ with $v_0 = s$ and $v_j = v$. Since we have topologically sorted the graph, we process the edges in order $(v_0, v_1), (v_1, v_2), ..., (v_{j-1}, v_j)$ since each edge in the path is one farther from s than its preceding edge in the path. Thus by the path relaxation property, $d[v] = \delta(s, v)$.