Lecture 15: All-pairs shortest paths

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

November 2, 2023

Overview

- Single-Source Shortest Paths with Arbitrary Weights
  - How Dijkstra goes wrong
  - Negative Cycles
  - Bellman-Ford: Dynamic Programming for the rescue!
  - Shortest Path Tree
- All-Pairs Shortest Paths
  - Floyd-Warshall: "mo' paths mo' subproblems"
- Acknowledgements

Single-Source Shortest Paths

- **Input: Input:** *Weighted directed* graph $G(V, E, w)$, where $w : E \to \mathbb{R}$, vertex $s \in V$

  Adjacency list. Note that weights can be arbitrary.

- **Output:** a shortest path from s to t for any $t \in V$

Single-Source Shortest Paths

- **Input: Input:** *Weighted directed* graph $G(V, E, w)$, where $w : E \to \mathbb{R}$, vertex $s \in V$

  Adjacency list. Note that weights can be arbitrary.
- **Output:** a shortest path from s to t for any $t \in V$
- Why not Dijkstra?
- Negative weights $\Rightarrow$ may have negative cycles.

Single-Source Shortest Paths

- **Input: Input:** *Weighted directed* graph $G(V, E, w)$, where $w : E \to \mathbb{R}$, vertex $s \in V$

  Adjacency list. Note that weights can be arbitrary.
- **Output:** a shortest path from s to t for any $t \in V$
- Why not Dijkstra?
- Negative weights $\Rightarrow$ may have negative cycles.
- Even without negative cycles Dijkstra will fail, since we violate the property of the distance!

Negative Cycles

- With negative edges, we may have negative cycles
- In this case, "shortest paths" will have length $-\infty$ (i.e., not well defined)

Negative Cycles

- With negative edges, we may have negative cycles
- In this case, "shortest paths" will have length $-\infty$ (i.e., not well defined)
- If we have no negative cycles, then shortest path distance well defined (as cycles don't help you "go faster")

Negative Cycles

- With negative edges, we may have negative cycles
- In this case, "shortest paths" will have length $-\infty$ (i.e., not well defined)
- If we have no negative cycles, then shortest path distance well defined (as cycles don't help you "go faster")
- Can we devise an efficient algorithm that solves the following problem:
  - If G has a negative cycle, output FAIL (output the cycle)
  - Else, solve the single-source shortest paths problem

- Single-Source Shortest Paths with Arbitrary Weights
  - How Dijkstra goes wrong
  - Negative Cycles
  - Bellman-Ford: Dynamic Programming for the rescue!
  - Shortest Path Tree
- All-Pairs Shortest Paths
  - Floyd-Warshall: “mo’ paths mo’ subproblems”
- Acknowledgements

Bellman-Ford

- **Template:**
  1. *Subproblems*

     $D[v, i] :=$ shortest $s \to v$ path distance using at most i edges
  2. *Base case*: $D[s, 0] = 0$, and $D[v, 0] = \infty$, for all $v \neq s$
  3. *Recurrence*:

     $$D[v, i+1] = \min \left\{ D[v, i], \min_{u \in N_{in}(v)} (D[u, i] + w(u, v)) \right\}$$
  4. **Output:** $D[v, n-1]$ for all $v \in V$

Bellman-Ford

- **Template:**
  1. *Subproblems*

     $$D[v, i] := \text{shortest } s \to v \text{ path distance using at most } i \text{ edges}$$
  2. *Base case*: $D[s, 0] = 0$, and $D[v, 0] = \infty$, for all $v \neq s$
  3. *Recurrence*:

     $$D[v, i+1] = \min \left\{ D[v, i], \min_{u \in N_{in}(v)} (D[u, i] + w(u, v)) \right\}$$
  4. **Output:** $D[v, n-1]$ for all $v \in V$
- Why do we only need to check $n - 1$ times?

  If graph has no negative cycle, then shortest walk must be simple path $\Rightarrow \ \leq n - 1$ edges.

Bellman-Ford

- **Algorithm:**
  1. **Initialization:** $D[s,i] = 0,\ D[v,i] = \infty$ for all $v \neq s$ and $i \in [0, n-1]$
     $p[v] = NULL$ for all $v \in V$
  2. for $i \in [n-1]$:
     for $(u,v) \in E$:
     if $D[u,i-1] + w(u,v) < D[v,i]$:
     $D[v,i+1] = D[v,i] = D[u,i-1] + w(u,v)$
     $p[v] = u$
  3. **Output:** $D[v, n-1]$ for all $v \in V$

Bellman-Ford

- **Algorithm:**
  1. **Initialization:** $D[s,i] = 0$, $D[v,i] = \infty$ for all $v \neq s$ and $i \in [0, n-1]$
     $p[v] = NULL$ for all $v \in V$
  2. for $i \in [n-1]$:
     for $(u,v) \in E$:
     if $D[u, i-1] + w(u,v) < D[v,i]$:
     $D[v, i+1] = D[v,i] = D[u, i-1] + w(u,v)$
     $p[v] = u$
  3. **Output:** $D[v, n-1]$ for all $v \in V$
- **Running time:**
  - Initialization: $O(n)$
  - First for loop runs for $O(n)$ iterations. Each iteration takes $O(m)$ operations, as the for loop on edges takes $O(m)$.
  - Total: $O(nm)$

Bellman-Ford

- **Algorithm:**
  1. **Initialization:** $D[s,i] = 0$, $D[v,i] = \infty$ for all $v \neq s$ and $i \in [0, n-1]$
     $p[v] = NULL$ for all $v \in V$
  2. for $i \in [n-1]$:
     for $(u,v) \in E$:
     if $D[u,i-1] + w(u,v) < D[v,i]$:
     $D[v,i+1] = D[v,i] = D[u,i-1] + w(u,v)$
     $p[v] = u$
  3. **Output:** $D[v, n-1]$ for all $v \in V$
- **Running time:**
  - Initialization: $O(n)$
  - First for loop runs for $O(n)$ iterations. Each iteration takes $O(m)$ operations, as the for loop on edges takes $O(m)$.
  - Total: $O(nm)$
- **Space Complexity:** $O(n^2)$

Bellman-Ford

- **Algorithm:**
  1. **Initialization:** $D[s, i] = 0$, $D[v, i] = \infty$ for all $v \neq s$ and $i \in [0, n-1]$
     $p[v] = NULL$ for all $v \in V$
  2. for $i \in [n-1]$:
     for $(u, v) \in E$:
     if $D[u, i-1] + w(u, v) < D[v, i]$:
     $D[v, i+1] = D[v, i] = D[u, i-1] + w(u, v)$
     $p[v] = u$
  3. **Output:** $D[v, n-1]$ for all $v \in V$
- **Running time:**
  - Initialization: $O(n)$
  - First for loop runs for $O(n)$ iterations. Each iteration takes $O(m)$ operations, as the for loop on edges takes $O(m)$.
  - Total: $O(nm)$
- **Space Complexity:** $O(n^2)$
- Can reduce space used to be $O(n)$ by not keeping track of i (exercise)

Shortest Path Tree

- As in BFS and Dijkstra, would like to return *succinct* data structure will all shortest paths

Shortest Path Tree

- As in BFS and Dijkstra, would like to return *succinct* data structure will all shortest paths
- Bellman-Ford has many iterations, not clear whether edges $(p[u], u)$ will form a tree.

Shortest Path Tree

- As in BFS and Dijkstra, would like to return *succinct* data structure will all shortest paths
- Bellman-Ford has many iterations, not clear whether edges $(p[u], u)$ will form a tree.
- However, if we have a cycle "it must be making some path shorter" which means it must be a negative cycle!

Lemma (Negative Cycles)

If there is a directed cycle Γ in the parent subgraph given by $(p[u], u)$, then Γ must be a negative cycle.

Shortest Path Tree

- However, if we have a cycle "it must be making some path shorter" which means it must be a negative cycle!

Lemma (Negative Cycles)

If there is a directed cycle Γ in the parent subgraph given by $(p[u], u)$, then Γ must be a negative cycle.

- Let $\Gamma = (v_1, v_2, \cdots, v_k)$, completed at path length i.

Shortest Path Tree

- However, if we have a cycle "it must be making some path shorter" which means it must be a negative cycle!

Lemma (Negative Cycles)

If there is a directed cycle Γ in the parent subgraph given by $(p[u], u)$, then Γ must be a negative cycle.

- Let $\Gamma = (v_1, v_2, \cdots, v_k)$, completed at path length i.
- For $1 \leq j < k$ must have

$$D[v_{j+1}, i] = D[v_j, i'_j] + w(v_j, v_{j+1})$$

for some $i'_j < i$ as $p[v_{i+1}] = v_i$.

Shortest Path Tree

- However, if we have a cycle "it must be making some path shorter" which means it must be a negative cycle!

Lemma (Negative Cycles)

If there is a directed cycle Γ in the parent subgraph given by $(p[u], u)$, then Γ must be a negative cycle.

- Let $\Gamma = (v_1, v_2, \cdots, v_k)$, completed at path length i.
- For $1 \le j < k$ must have

$$D[v_{j+1}, i] = D[v_j, i_j'] + w(v_j, v_{j+1})$$

for some $i_j' < i$ as $p[v_{i+1}] = v_i$.
- Since distances only decrease as path length increases, by our algorithm, we have $D[v_j, i_j'] \le D[v_j, i-1] \le D[v_j, i]$ and thus

$$D[v_{j+1}, i] \ge D[v_j, i-1] + w(v_j, v_{j+1})$$

Shortest Path Tree

- However, if we have a cycle "it must be making some path shorter" which means it must be a negative cycle!

Lemma (Negative Cycles)

If there is a directed cycle Γ in the parent subgraph given by $(p[u], u)$, then Γ must be a negative cycle.

- Let $\Gamma = (v_1, v_2, \cdots, v_k)$, completed at path length i.
- For $1 \leq j < k$ must have

$$D[v_{j+1}, i] = D[v_j, i'_j] + w(v_j, v_{j+1})$$

for some $i'_j < i$ as $p[v_{i+1}] = v_i$.
- Since distances only decrease as path length increases, by our algorithm, we have $D[v_j, i'_j] \leq D[v_j, i-1] \leq D[v_j, i]$ and thus

$$D[v_{j+1}, i] \geq D[v_j, i-1] + w(v_j, v_{j+1})$$

- As cycle formed at length i we have to update $D[v_1, i]$, which means

$$D[v_1, i-1] > D[v_k, i-1] + w(v_k, v_1) \geq D[v_k, i] + w(v_k, v_1)$$

Shortest Path Tree

- Summing up inequalities, we have:

$$D[v_1, i-1] + \sum_{j=1}^{k-1} D[v_{j+1}, i] > \sum_{j=1}^{k} D[v_j, i-1] + \sum_{i=1}^{k-1} w(v_j, v_{j+1}) + w(v_k, v_1)$$

which implies

$$\sum_{j=2}^{k} D[v_j, i] > \sum_{j=2}^{k} D[v_j, i-1] + \sum_{i=1}^{k-1} w(v_j, v_{j+1}) + w(v_k, v_1)$$

Shortest Path Tree

- Summing up inequalities, we have:

$$D[v_1, i-1] + \sum_{j=1}^{k-1} D[v_{j+1}, i] > \sum_{j=1}^{k} D[v_j, i-1] + \sum_{i=1}^{k-1} w(v_j, v_{j+1}) + w(v_k, v_1)$$

  which implies

$$\sum_{j=2}^{k} D[v_j, i] > \sum_{j=2}^{k} D[v_j, i-1] + \sum_{i=1}^{k-1} w(v_j, v_{j+1}) + w(v_k, v_1)$$

- Using the fact that $D[v_j, i-1] \geq D[v_j, i]$, we obtain

$$\sum_{i=1}^{k-1} w(v_j, v_{j+1}) + w(v_k, v_1) < 0$$

Finding Negative Cycles

- So, how is Bellman-Ford finding a negative cycle, when it exists?

Finding Negative Cycles

- So, how is Bellman-Ford finding a negative cycle, when it exists?
- Need to prove that after $n-1$ iterations, we will find a negative cycle, if one exists (otherwise we already know our algorithm is correct)

Finding Negative Cycles

- So, how is Bellman-Ford finding a negative cycle, when it exists?
- Need to prove that after $n-1$ iterations, we will find a negative cycle, if one exists (otherwise we already know our algorithm is correct)
- If we have negative cycles, expect $D[v,k] \to -\infty$ as $k \to \infty$

  Will go through negative cycle multiple times.

Finding Negative Cycles

- So, how is Bellman-Ford finding a negative cycle, when it exists?
- Need to prove that after $n-1$ iterations, we will find a negative cycle, if one exists (otherwise we already know our algorithm is correct)
- If we have negative cycles, expect $D[v,k] \to -\infty$ as $k \to \infty$

  Will go through negative cycle multiple times.
- If we have no negative cycles, then $D[v,n] = D[v,n-1]$ for all $v \in V$

Finding Negative Cycles

- So, how is Bellman-Ford finding a negative cycle, when it exists?
- Need to prove that after $n-1$ iterations, we will find a negative cycle, if one exists (otherwise we already know our algorithm is correct)
- If we have negative cycles, expect $D[v,k] \to -\infty$ as $k \to \infty$

  Will go through negative cycle multiple times.
- If we have no negative cycles, then $D[v,n] = D[v,n-1]$ for all $v \in V$
- Thus, to compute negative cycles, just need to check if $D[v,n-1] = D[v,n]$

Negative Cycle Lemmas

Lemma

If G has negative cycle in SCC containing s, then for some $v \in V$,

$$D[v, k] \to -\infty \text{ as } k \to \infty$$

Negative Cycle Lemmas

Lemma

If G has negative cycle in SCC containing s, then for some $v \in V$,

$$D[v, k] \to -\infty \text{ as } k \to \infty$$

Proof: by definition of $D[v, k]$ and our recurrence, we are always computing the min distance of $s \to v$ paths of length k. Since we can take the negative cycle multiple times as $k \to \infty$ there will always be a path of smaller length.

Negative Cycle Lemmas

Lemma

If G has negative cycle in SCC containing s, then for some $v \in V$,

$$D[v, k] \to -\infty \text{ as } k \to \infty$$

Lemma

If graph has no negative cycles, then

$$D[v, n] = D[v, n-1], \text{ for all } v \in V.$$

Negative Cycle Lemmas

Lemma

If G has negative cycle in SCC containing s, then for some $v \in V$,

$$D[v, k] \to -\infty \text{ as } k \to \infty$$

Lemma

If graph has no negative cycles, then

$$D[v, n] = D[v, n-1], \text{ for all } v \in V.$$

Proof: every cycle is non-negative, so it doesn't help. Any walk with length n must contain a cycle, thus not optimal. So we won't update D in the n^{th} iteration.

Negative Cycle Lemmas

Lemma

If G has negative cycle in SCC containing s, then for some $v \in V$,

$$D[v,k] \to -\infty \text{ as } k \to \infty$$

Lemma

If $D[v,n] = D[v,n-1]$ for all $v \in V$, then G has no negative cycle.

Proof: By recurrence, and the assumption, we will prove that $D[v, n+t] = D[v, n-1]$ for all $t \geq 0$. Then first lemma will finish it.

$$\begin{aligned} D[v, n+t+1] &= \min\left\{ D[v, n+t], \min_{u \in N_{in}(v)} (D[v, n+t] + w(u,v)) \right\} \\ &= \min\left\{ D[v, n-1], \min_{u \in N_{in}(v)} (D[v, n-1] + w(u,v)) \right\} \\ &= D[v,n] = D[v, n-1] \end{aligned}$$

Since no distance $\to -\infty$, G has no negative cycles.

Full Bellman-Ford

- **Algorithm:**
  1. **Initialization:** $D[s,i] = 0$, $D[v,i] = \infty$ for all $v \neq s$ and $i \in [0,n]$
     $p[v] = NULL$ for all $v \in V$
  2. for $i \in [n]$:
     for $(u,v) \in E$:
     if $D[u,i-1] + w(u,v) < D[v,i]$:
     $D[v,i+1] = D[v,i] = D[u,i-1] + w(u,v)$
     $p[v] = u$
  3. If $D[v,n-1] = D[v,n]$ for all $v \in V$ **output:** $D[v,n-1]$ for all $v \in V$
  4. Else, **output** FAIL.

Full Bellman-Ford

- **Algorithm:**
  1. **Initialization:** $D[s,i] = 0$, $D[v,i] = \infty$ for all $v \neq s$ and $i \in [0,n]$
     $p[v] = NULL$ for all $v \in V$
  2. for $i \in [n]$:
     for $(u,v) \in E$:
     if $D[u,i-1] + w(u,v) < D[v,i]$:
     $D[v,i+1] = D[v,i] = D[u,i-1] + w(u,v)$
     $p[v] = u$
  3. If $D[v,n-1] = D[v,n]$ for all $v \in V$ **output:** $D[v,n-1]$ for all $v \in V$
  4. Else, **output** FAIL.
- same running time as before, and space complexity

Full Bellman-Ford

- **Algorithm:**
  1. **Initialization:** $D[s, i] = 0$, $D[v, i] = \infty$ for all $v \neq s$ and $i \in [0, n]$
     $p[v] = NULL$ for all $v \in V$
  2. for $i \in [n]$:
     for $(u, v) \in E$:
     if $D[u, i-1] + w(u, v) < D[v, i]$:
     $D[v, i+1] = D[v, i] = D[u, i-1] + w(u, v)$
     $p[v] = u$
  3. If $D[v, n-1] = D[v, n]$ for all $v \in V$ **output:** $D[v, n-1]$ for all $v \in V$
  4. Else, **output** FAIL.
- same running time as before, and space complexity
- easy to find negative cycle (exercise)

- Single-Source Shortest Paths with Arbitrary Weights
  - How Dijkstra goes wrong
  - Negative Cycles
  - Bellman-Ford: Dynamic Programming for the rescue!
  - Shortest Path Tree
- All-Pairs Shortest Paths
  - Floyd-Warshall: “mo’ paths mo’ subproblems”
- Acknowledgements

All-Pairs Shortest Paths

- **Input:** Directed, weighted graph $G(V, E, w)$, without negative cycles
- **Output:** for all pairs (u, v), the length of the shortest $u \to v$ path

All-Pairs Shortest Paths

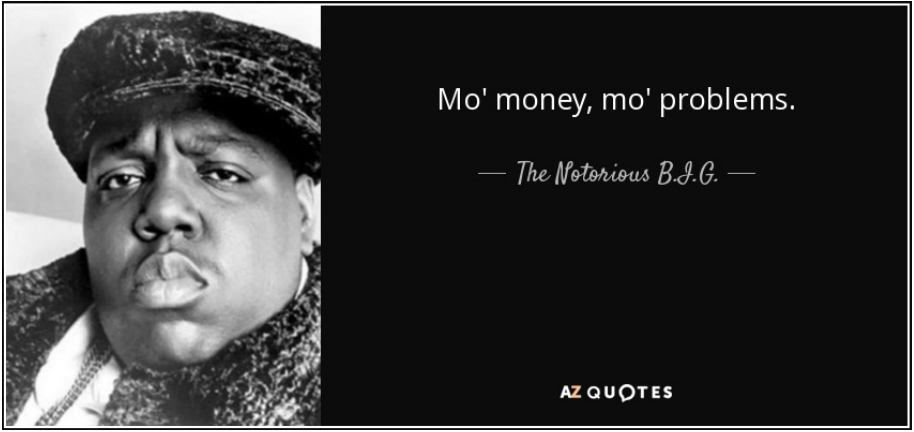
- **Input:** Directed, weighted graph $G(V, E, w)$, without negative cycles
- **Output:** for all pairs (u, v), the length of the shortest $u \to v$ path
- Could apply n iterations of Bellman-Ford, but this would give us $O(n^2 m)$ running time

All-Pairs Shortest Paths

- **Input:** Directed, weighted graph $G(V, E, w)$, without negative cycles
- **Output:** for all pairs (u, v), the length of the shortest $u \to v$ path
- Could apply n iterations of Bellman-Ford, but this would give us $O(n^2 m)$ running time
- Can we do better?

Floyd-Warshall: $O(n^3)$

Floyd-Warshall



Floyd-Warshall

- Simple modification of Bellman-Ford to account for all sources
- **Template:**
  1. *Subproblems*

     $D[u, v, k] :=$ shortest $u \to v$ path distance using only vertices $\{1, 2, \ldots, k\}$ as *intermediate* vertices.
  2. *Base case*: $D[u, v, 0] = w(u, v)$, if $(u, v) \in E$ and $D[u, v, 0] = \infty$, otherwise.
  3. *Recurrence*:

     $$D[u, v, k+1] = \min\{D[u, v, k],\ D[u, k+1, k] + D[k+1, v, k]\}$$
  4. **Output:** $D[u, v, n]$ for all $u, v \in V$

Full Floyd-Warshall Algorithm

- **Algorithm:**
  1. **Initialize:** $D[u, v, 0] = w(u, v)$, if $(u, v) \in E$ and $D[u, v, 0] = \infty$, otherwise.
  2. for $k \in [n]$:
     for $u \in [n]$:
     for $v \in [n]$:
     Compute recurrence

$$D[u, v, k] = \min\{D[u, v, k-1],\ D[u, k, k-1] + D[k, v, k-1])\}$$

  3. Return $D[u, v, n]$

Full Floyd-Warshall Algorithm

- **Algorithm:**
  1. **Initialize:** $D[u, v, 0] = w(u, v)$, if $(u, v) \in E$ and $D[u, v, 0] = \infty$, otherwise.
  2. for $k \in [n]$:
     for $u \in [n]$:
     for $v \in [n]$:
     Compute recurrence

$$D[u, v, k] = \min\{D[u, v, k-1],\ D[u, k, k-1] + D[k, v, k-1])\}$$

  3. Return $D[u, v, n]$
- **Running Time:** three nested loops, each of length n. Computing within the loops takes $O(1)$ time, so total running time $O(n^3)$.

Full Floyd-Warshall Algorithm

- **Algorithm:**
  1. **Initialize:** $D[u, v, 0] = w(u, v)$, if $(u, v) \in E$ and $D[u, v, 0] = \infty$, otherwise.
  2. for $k \in [n]$:
     for $u \in [n]$:
     for $v \in [n]$:
     Compute recurrence

$$D[u, v, k] = \min\{D[u, v, k-1],\ D[u, k, k-1] + D[k, v, k-1]\}$$

  3. Return $D[u, v, n]$
- **Running Time:** three nested loops, each of length n. Computing within the loops takes $O(1)$ time, so total running time $O(n^3)$.
- **Correctness:** follows from correctness of recurrence, and the fact that we have precomputed correctly.

Acknowledgement

- Based on Prof. Lau's Lecture 14

  `https://cs.uwaterloo.ca/~lapchi/cs341/notes/L14.pdf`

References I

Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford. (2009)

Introduction to Algorithms, third edition.

MIT Press

Kleinberg, Jon and Tardos, Eva (2006)

Algorithm Design.

Addison Wesley