All Pairs Shortest Paths

Eric Price

UT Austin

CS 331, Spring 2020 Coronavirus Edition

Talk Outline

1. APSP
2. Problems

All Pairs Shortest Paths

- Given a graph G, find shortest $s \rightsquigarrow t$ path distance for *all* $s, t \in V$.

All Pairs Shortest Paths

- Given a graph G, find shortest $s \rightsquigarrow t$ path distance for *all* $s, t \in V$.
- Approaches:

All Pairs Shortest Paths

- Given a graph G, find shortest $s \rightsquigarrow t$ path distance for *all* $s, t \in V$.
- Approaches:
  - Bellman-Ford for all s: $O(V^2 E)$

All Pairs Shortest Paths

- Given a graph G, find shortest $s \rightsquigarrow t$ path distance for *all* $s, t \in V$.
- Approaches:
  - Bellman-Ford for all s: $O(V^2E)$
  - Dijkstra for all s: $O(VE + V^2 \log V)$ if nonnegative weights

All Pairs Shortest Paths

- Given a graph G, find shortest $s \rightsquigarrow t$ path distance for *all* $s, t \in V$.
- Approaches:
  - Bellman-Ford for all s: $O(V^2 E)$
  - Dijkstra for all s: $O(VE + V^2 \log V)$ if nonnegative weights
  - Floyd-Warshall: $O(V^3)$

All Pairs Shortest Paths

- Given a graph G, find shortest $s \rightsquigarrow t$ path distance for *all* $s, t \in V$.
- Approaches:
  - Bellman-Ford for all s: $O(V^2E)$
  - Dijkstra for all s: $O(VE + V^2 \log V)$ if nonnegative weights
  - Floyd-Warshall: $O(V^3)$
  - Johnson: $O(VE + V^2 \log V)$ in general

Floyd-Warshall

function $\textsc{FloydWarshall}(A)$ $\qquad \triangleright\ A[u \to v]$ is cost of $u \to v$ edge

  $D \leftarrow A$ $\qquad \triangleright$ (or ∞ if no edge)

  for $w \in V$ **do**

    for $u \in V$ **do**

      for $v \in V$ **do**

        $D[u, v] \leftarrow \min(D[u, v], D[u, w] + D[w, v])$

  return D

Floyd-Warshall

function $\textsc{FloydWarshall}(A)$ $\qquad \triangleright$ $A[u \to v]$ is cost of $u \to v$ edge

$\quad D \leftarrow A$ $\qquad \triangleright$ (or ∞ if no edge)

$\quad$ **for** $w \in V$ **do**

$\quad\quad$ **for** $u \in V$ **do**

$\quad\quad\quad$ **for** $v \in V$ **do**

$\quad\quad\quad\quad D[u, v] \leftarrow \min(D[u, v], D[u, w] + D[w, v])$

$\quad$ **return** D

- Takes an adjacency matrix, returns a distance matrix

Floyd-Warshall

function FLOYDWARSHALL(A) $\triangleright$ $A[u \to v]$ is cost of $u \to v$ edge
    $D \leftarrow A$ $\triangleright$ (or ∞ if no edge)
    for $w \in V$ **do**
        for $u \in V$ **do**
            for $v \in V$ **do**
                $D[u, v] \leftarrow \min(D[u, v], D[u, w] + D[w, v])$
    return D

- Takes an adjacency matrix, returns a distance matrix
- $O(n^3)$ time

Floyd-Warshall

```
function FloydWarshall(A)        ▷ A[u → v] is cost of u → v edge
    D ← A                                   ▷ (or ∞ if no edge)
    for w ∈ V do
        for u ∈ V do
            for v ∈ V do
                D[u, v] ← min(D[u, v], D[u, w] + D[w, v])
    return D
```

- Takes an adjacency matrix, returns a distance matrix
- $O(n^3)$ time

Lemma

Let $P = (s, u_1, \ldots, u_k, t)$ be a shortest $s \to t$ path. After visiting $w \notin \{s, t\}$, $P \setminus \{w\}$ is also a shortest $s \rightsquigarrow t$ path in D.

Floyd-Warshall

function $\textsc{FloydWarshall}(A)$ ▷ $A[u \to v]$ is cost of $u \to v$ edge
    $D \leftarrow A$ ▷ (or ∞ if no edge)
    for $w \in V$ **do**
        for $u \in V$ **do**
            for $v \in V$ **do**
                $D[u, v] \leftarrow \min(D[u, v], D[u, w] + D[w, v])$
    return D

- Takes an adjacency matrix, returns a distance matrix
- $O(n^3)$ time

Lemma

Let $P = (s, u_1, \ldots, u_k, t)$ be a shortest $s \to t$ path. After visiting $w \notin \{s, t\}$, $P \setminus \{w\}$ is also a shortest $s \rightsquigarrow t$ path in D.

- Q: negative edges?

Floyd-Warshall

```
function FloydWarshall(A)          ▷ A[u → v] is cost of u → v edge
    D ← A                          ▷ (or ∞ if no edge)
    for w ∈ V do
        for u ∈ V do
            for v ∈ V do
                D[u, v] ← min(D[u, v], D[u, w] + D[w, v])
    return D
```

- Takes an adjacency matrix, returns a distance matrix
- $O(n^3)$ time

Lemma

Let $P = (s, u_1, \ldots, u_k, t)$ be a shortest $s \to t$ path. After visiting $w \notin \{s, t\}$, $P \setminus \{w\}$ is also a shortest $s \rightsquigarrow t$ path in D.

- Q: negative edges? OK

Floyd-Warshall

function $\textsc{FloydWarshall}(A)$ $\qquad \triangleright$ $A[u \to v]$ is cost of $u \to v$ edge
  $D \leftarrow A$ $\qquad \triangleright$ (or ∞ if no edge)
  for $w \in V$ **do**
    for $u \in V$ **do**
      for $v \in V$ **do**
        $D[u, v] \leftarrow \min(D[u, v], D[u, w] + D[w, v])$
  return D

- Takes an adjacency matrix, returns a distance matrix
- $O(n^3)$ time

Lemma

Let $P = (s, u_1, \ldots, u_k, t)$ be a shortest $s \to t$ path. After visiting $w \notin \{s, t\}$, $P \setminus \{w\}$ is also a shortest $s \rightsquigarrow t$ path in D.

- Q: negative edges? OK Negative cycles?

Floyd-Warshall

```
function FloydWarshall(A)                      ▷ A[u → v] is cost of u → v edge
    D ← A                                      ▷ (or ∞ if no edge)
    for w ∈ V do
        for u ∈ V do
            for v ∈ V do
                D[u, v] ← min(D[u, v], D[u, w] + D[w, v])
    return D
```

- Takes an adjacency matrix, returns a distance matrix
- $O(n^3)$ time

Lemma

Let $P = (s, u_1, \ldots, u_k, t)$ be a shortest $s \to t$ path. After visiting $w \notin \{s, t\}$, $P \setminus \{w\}$ is also a shortest $s \rightsquigarrow t$ path in D.

- Q: negative edges? OK Negative cycles? Check if diagonal < 0

Floyd-Warshall

function $\textsc{FloydWarshall}(A)$ $\quad \triangleright$ $A[u \to v]$ is cost of $u \to v$ edge
  $D \leftarrow A$ $\quad \triangleright$ (or ∞ if no edge)
  for $w \in V$ **do**
    for $u \in V$ **do**
      for $v \in V$ **do**
        $D[u, v] \leftarrow \min(D[u, v], D[u, w] + D[w, v])$
  return D

- Takes an adjacency matrix, returns a distance matrix
- $O(n^3)$ time

Lemma

Let $P = (s, u_1, \ldots, u_k, t)$ be a shortest $s \to t$ path. After visiting $w \notin \{s, t\}$, $P \setminus \{w\}$ is also a shortest $s \rightsquigarrow t$ path in D.

- Q: negative edges? OK Negative cycles? Check if diagonal < 0

Johnson's algorithm

- Recall:

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

    has shortest path distances

$$D'[s, t] = D[s, t] + h(t) - h(s).$$

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

has shortest path distances

$$D'[s,t] = D[s,t] + h(t) - h(s).$$

- Idea: in $O(VE + V^2 \log V)$,

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

has shortest path distances

$$D'[s,t] = D[s,t] + h(t) - h(s).$$

- Idea: in $O(VE + V^2 \log V)$,
  1. Compute a single h so $w'(u \to v) \geq 0$ for all h.

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

    $$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

    has shortest path distances

    $$D'[s, t] = D[s, t] + h(t) - h(s).$$

- Idea: in $O(VE + V^2 \log V)$,
  1. Compute a single h so $w'(u \to v) \geq 0$ for all h. (h consistent)

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

    $$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

    has shortest path distances

    $$D'[s,t] = D[s,t] + h(t) - h(s).$$

- Idea: in $O(VE + V^2 \log V)$,
  1. Compute a single h so $w'(u \to v) \geq 0$ for all h. (h consistent part 2)

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

    $$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

    has shortest path distances

    $$D'[s,t] = D[s,t] + h(t) - h(s).$$

- Idea: in $O(VE + V^2 \log V)$,
  1. Compute a single h so $w'(u \to v) \geq 0$ for all h. (h consistent part 2)
  2. Compute D' for every source s using Dijkstra on w'.

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

    $$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

    has shortest path distances

    $$D'[s,t] = D[s,t] + h(t) - h(s).$$

- Idea: in $O(VE + V^2 \log V)$,
  1. Compute a single h so $w'(u \to v) \geq 0$ for all h. (h consistent part 2)
  2. Compute D' for every source s using Dijkstra on w'.
  3. Output D.

Johnson's algorithm

- Recall:
  - Dijkstra would be great if we had nonnegative edges
  - Reweighting: for any $h : V \to \mathbb{R}$, the graph with weights

    $$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

    has shortest path distances

    $$D'[s,t] = D[s,t] + h(t) - h(s).$$

- Idea: in $O(VE + V^2 \log V)$,
  1. Compute a single h so $w'(u \to v) \geq 0$ for all h. (h consistent part 2)
  2. Compute D' for every source s using Dijkstra on w'.
  3. Output D.

Finding a good heuristic

- Pick an arbitrary source s^*.

Finding a good heuristic

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.

Finding a good heuristic

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.

Finding a good heuristic

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$

Finding a good heuristic

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set

$$h(u) := -D[s^*, u]$$

- Now, for any $u, v \in V$,

$$\begin{aligned} w'(u \to v) &:= w(u \to v) - h(u) + h(v) \\ &= w(u \to v) + D[s^*, u] - D[s^*, v]. \end{aligned}$$

Finding a good heuristic

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$
- Now, for any $u, v \in V$,
$$\begin{aligned} w'(u \to v) &:= w(u \to v) - h(u) + h(v) \\ &= w(u \to v) + D[s^*, u] - D[s^*, v]. \end{aligned}$$

  But by the triangle inequality,
$$D[s^*, v] \leq D[s^*, u] + w(u \to v)$$
  so $w'(u \to v) \geq 0$.

Johnson's algorithm

- Pick an arbitrary source s^*.

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$
- Define the graph with weights
$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$
- Define the graph with weights
$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$
- For every $s \in V$, run Dijkstra from s on w' to compute distance matrix D'

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$
- Define the graph with weights
$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$
- For every $s \in V$, run Dijkstra from s on w' to compute distance matrix D'
- Output
$$D[s, t] = D'[s, t] - h(t) + h(s).$$

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$
- Define the graph with weights
$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$
- For every $s \in V$, run Dijkstra from s on w' to compute distance matrix D'
- Output
$$D[s, t] = D'[s, t] - h(t) + h(s).$$
- Overall: $O(VE + V^2 \log V)$ time.

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$
- Define the graph with weights
$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$
- For every $s \in V$, run Dijkstra from s on w' to compute distance matrix D'
- Output
$$D[s, t] = D'[s, t] - h(t) + h(s).$$
- Overall: $O(VE + V^2 \log V)$ time.
- Q:

Johnson's algorithm

- Pick an arbitrary source s^*.
- Compute shortest paths $D[s^*, u]$ from s^* on original graph.
  - with Bellman-Ford in $O(VE)$ time.
- Set
$$h(u) := -D[s^*, u]$$
- Define the graph with weights
$$w'(u \to v) := w(u \to v) - h(u) + h(v).$$
- For every $s \in V$, run Dijkstra from s on w' to compute distance matrix D'
- Output
$$D[s, t] = D'[s, t] - h(t) + h(s).$$
- Overall: $O(VE + V^2 \log V)$ time.
- Q: negative cycles?

All Pairs Shortest Paths

- Given a graph G, find shortest $s \rightsquigarrow t$ path distance for *all* $s, t \in V$.
- Approaches:
  - Bellman-Ford for all s: $O(V^2 E)$
  - Dijkstra for all s: $O(VE + V^2 \log V)$ if nonnegative weights
  - Floyd-Warshall: $O(V^3)$
  - Johnson: $O(VE + V^2 \log V)$ in general

Talk Outline

Shortest Path Problems

http://jeffe.cs.illinois.edu/teaching/algorithms/book/08-sssp.pdf

- Problem 2: Dijkstra with k negative edges.
- Problem 3: *vertices*, not edges, have weight.
- Problem 5: edge reinsertion
- Problem 4: Replacement paths on directed graphs
- Problem 12: Smallest shortest path
- Problem 16, 17: Remember reductions?
- Problem 1 of https://www.cs.utexas.edu/~ecprice/courses/331h/psets/331h-ps6.pdf