All-Pairs Shortest Paths

Tao Hou

Why study it?

We can solve an all-pairs shortest-paths problem by running a single-source shortest-paths algorithm ***for each vertex***:

- Use Dijkstra's algorithm: $O(|V||E| \log(|V|))$
  - For sparse graph, $|E| = \Theta(|V|)$: $O(|V|^2 \log(|V|))$ (not too bad)
  - For dense graph, $|E| = \Theta(|V|^2)$: $O(|V|^3 \log(|V|))$ (can do better)
  - Not to mention that edge weights have to be non-negative

We can solve an all-pairs shortest-paths problem by running a single-source shortest-paths algorithm ***for each vertex***:

- Use Dijkstra's algorithm: $O(|V||E|\log(|V|))$
  - For sparse graph, $|E| = \Theta(|V|)$: $O(|V|^2\log(|V|))$ (not too bad)
  - For dense graph, $|E| = \Theta(|V|^2)$: $O(|V|^3\log(|V|))$ (can do better)
  - Not to mention that edge weights have to be non-negative
- If we want to allow negative weights, we have to use Bellman-Ford
  - Time complexity: $O(|V|^2|E|)$
  - $O(|V|^4)$ for dense graphs (probably bad)

Why study it?

We can solve an all-pairs shortest-paths problem by running a single-source shortest-paths algorithm ***for each vertex***:

- Use Dijkstra's algorithm: $O(|V||E|\log(|V|))$
  - For sparse graph, $|E| = \Theta(|V|)$: $O(|V|^2\log(|V|))$ (not too bad)
  - For dense graph, $|E| = \Theta(|V|^2)$: $O(|V|^3\log(|V|))$ (can do better)
  - Not to mention that edge weights have to be non-negative
- If we want to allow negative weights, we have to use Bellman-Ford
  - Time complexity: $O(|V|^2|E|)$
  - $O(|V|^4)$ for dense graphs (probably bad)

We introduce ***Floyd-Warshall*** algorithm:

- Run in $O(|V|^3)$ time and allow negative weights

Floyd-Warshall: Setting

- Assume that the vertices are numbered $1, 2, \ldots, n$ where $n = |V|$
- The input is an $n \times n$ matrix $W = (w_{i,j})$ representing the edge weights (an *augmentation* of adjacency matrix):

$$w_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \text{weight of edge } (i,j) & \text{if } i \neq j \text{ and } (i,j) \in E \\ \infty & \text{if } i \neq j \text{ and } (i,j) \notin E \end{cases}$$

- Allow negative-weight edges, but ***assume that the input graph contains no negative-weight cycles***

Floyd-Warshall: Setting

- Assume that the vertices are numbered $1, 2, \ldots, n$ where $n = |V|$
- The input is an $n \times n$ matrix $W = (w_{i,j})$ representing the edge weights (an *augmentation* of adjacency matrix):

$$w_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \text{weight of edge } (i,j) & \text{if } i \neq j \text{ and } (i,j) \in E \\ \infty & \text{if } i \neq j \text{ and } (i,j) \notin E \end{cases}$$

- Allow negative-weight edges, but ***assume that the input graph contains no negative-weight cycles***
- Returns an $n \times n$ matrix $D = (d_{i,j})$, where $d_{i,j} = \delta(i,j)$
- Also returns a ***predecessor matrix*** $\Pi = (\pi_{i,j})$

$$\pi_{i,j} = \begin{cases} \text{Nil} & i = j \text{ or no path from } i \text{ to } j \\ \text{Predecessor of } j \text{ on a shortest path from } i \text{ to } j & \text{otherwise} \end{cases}$$

  - i-th row of Π defines a shortest-paths tree rooted at i (the procedure to print a shortest path from i should be evident from previous contents)

Floyd-Warshall: DP Ingredients

- A dynamic-programming approach utilizing the ***optimal substructure property*** of shortest paths
- As can be imagined, the parameter of the *OPT* function contains:
  i and j, the start and end vertices

Floyd-Warshall: DP Ingredients

- A dynamic-programming approach utilizing the ***optimal substructure property*** of shortest paths
- As can be imagined, the parameter of the *OPT* function contains:
  i and j, the start and end vertices
- However, if your *OPT* contains only i, j, then:

$$d(i,j) = \min\{d(i,\ell) + d(\ell,j) \mid \ell \in V\}$$

- It would be nearly *impossible* to find a valid ***evaluation order***
  - There is no natural definition of ‘size’ for the problems $d(i,j)$: they are all ‘equal’; no one is a natural ‘subproblem’ of another
  - Also no natural ***base cases***

Floyd-Warshall: DP Ingredients

- The solution is that, we introduce ***another parameter*** k, and consider all paths from i to j whose ***intermediate vertices*** are $\leq k$
  - E.g., path $p = \langle v_1 = i, v_2, \ldots, v_{q-1}, v_q = j \rangle$ where $v_2, \ldots, v_{q-1} \leq k$

- The solution is that, we introduce ***another parameter*** k, and consider all paths from i to j whose ***intermediate vertices*** are $\leq k$
    - E.g., path $p = \langle v_1 = i, v_2, \ldots, v_{q-1}, v_q = j \rangle$ where $v_2, \ldots, v_{q-1} \leq k$
- ***OPT function***: Let $d_{i,j}^{(k)} := d(i, j, k)$ be the minimum weight of all paths from i to j with intermediate vertices $\leq k$
- We have the following immediate evidence why this definition makes sense:
  (1) We could easily identify the ***base case***: $d_{i,j}^{(0)} = w_{i,j}$
  (2) There is a natural notion of "size": k

Floyd-Warshall: DP Ingredients

- The solution is that, we introduce ***another parameter*** k, and consider all paths from i to j whose ***intermediate vertices*** are $\leq k$
  - E.g., path $p = \langle v_1 = i, v_2, \ldots, v_{q-1}, v_q = j \rangle$ where $v_2, \ldots, v_{q-1} \leq k$
- ***OPT function***: Let $d_{i,j}^{(k)} := d(i,j,k)$ be the minimum weight of all paths from i to j with intermediate vertices $\leq k$
- We have the following immediate evidence why this definition makes sense:
  (1) We could easily identify the ***base case***: $d_{i,j}^{(0)} = w_{i,j}$
  (2) There is a natural notion of "size": k
- Now consider defining $d_{i,j}^{(k)}$ for general k.
  Paths from i to j with intermediate vertices $\leq k$ fall in two sets:
  - Does not contain k:
  - Contains k:

Floyd-Warshall: DP Ingredients

- The solution is that, we introduce ***another parameter*** k, and consider all paths from i to j whose ***intermediate vertices*** are $\leq k$
  - E.g., path $p = \langle v_1 = i, v_2, \ldots, v_{q-1}, v_q = j \rangle$ where $v_2, \ldots, v_{q-1} \leq k$
- ***OPT function***: Let $d_{i,j}^{(k)} := d(i, j, k)$ be the minimum weight of all paths from i to j with intermediate vertices $\leq k$
- We have the following immediate evidence why this definition makes sense:
  (1) We could easily identify the ***base case***: $d_{i,j}^{(0)} = w_{i,j}$
  (2) There is a natural notion of "size": k
- Now consider defining $d_{i,j}^{(k)}$ for general k.
  Paths from i to j with intermediate vertices $\leq k$ fall in two sets:
  - Does not contain k: intermediate vertices are $\leq k - 1$; the shortest one is
  - Contains k:

Floyd-Warshall: DP Ingredients

- The solution is that, we introduce ***another parameter*** k, and consider all paths from i to j whose ***intermediate vertices*** are $\leq k$
  - E.g., path $p = \langle v_1 = i, v_2, \ldots, v_{q-1}, v_q = j \rangle$ where $v_2, \ldots, v_{q-1} \leq k$
- ***OPT function***: Let $d_{i,j}^{(k)} := d(i,j,k)$ be the minimum weight of all paths from i to j with intermediate vertices $\leq k$
- We have the following immediate evidence why this definition makes sense:
  (1) We could easily identify the ***base case***: $d_{i,j}^{(0)} = w_{i,j}$
  (2) There is a natural notion of "size": k
- Now consider defining $d_{i,j}^{(k)}$ for general k.
  Paths from i to j with intermediate vertices $\leq k$ fall in two sets:
  - Does not contain k: intermediate vertices are $\leq k-1$; the shortest one is $d_{i,j}^{(k-1)}$
  - Contains k:

Floyd-Warshall: DP Ingredients

- The solution is that, we introduce ***another parameter*** k, and consider all paths from i to j whose ***intermediate vertices*** are $\leq k$
  - E.g., path $p = \langle v_1 = i, v_2, \ldots, v_{q-1}, v_q = j \rangle$ where $v_2, \ldots, v_{q-1} \leq k$
- ***OPT function***: Let $d_{i,j}^{(k)} := d(i,j,k)$ be the minimum weight of all paths from i to j with intermediate vertices $\leq k$
- We have the following immediate evidence why this definition makes sense:
  (1) We could easily identify the ***base case***: $d_{i,j}^{(0)} = w_{i,j}$
  (2) There is a natural notion of “size”: k
- Now consider defining $d_{i,j}^{(k)}$ for general k.
  Paths from i to j with intermediate vertices $\leq k$ fall in two sets:
  - Does not contain k: intermediate vertices are $\leq k-1$; the shortest one is $d_{i,j}^{(k-1)}$
  - Contains k: the shortest one is $d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$

Floyd-Warshall: DP Ingredients

- The solution is that, we introduce ***another parameter*** k, and consider all paths from i to j whose ***intermediate vertices*** are $\leq k$
  - E.g., path $p = \langle v_1 = i, v_2, \ldots, v_{q-1}, v_q = j \rangle$ where $v_2, \ldots, v_{q-1} \leq k$
- ***OPT function***: Let $d_{i,j}^{(k)} := d(i,j,k)$ be the minimum weight of all paths from i to j with intermediate vertices $\leq k$
- We have the following immediate evidence why this definition makes sense:
  (1) We could easily identify the ***base case***: $d_{i,j}^{(0)} = w_{i,j}$
  (2) There is a natural notion of "size": k
- Now consider defining $d_{i,j}^{(k)}$ for general k.
  Paths from i to j with intermediate vertices $\leq k$ fall in two sets:
  - Does not contain k: intermediate vertices are $\leq k-1$; the shortest one is $d_{i,j}^{(k-1)}$
  - Contains k: the shortest one is $d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$
- So,

$$d_{i,j}^{(k)} = \begin{cases} w_{i,j} & k = 0 \\ \min\left\{d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}\right\} & k > 0 \end{cases}$$

Floyd-Warshall: Algorithm

FLOYD-WARSHALL(W)

$D^{(0)} = W$
for $k = 1, \ldots, n$
    $D^{(k)} := \left(d_{i,j}^{(k)}\right)$ be a new $n \times n$ matrix
    for $i = 1, \ldots, n$
        for $j = 1, \ldots, n$
            $d_{i,j}^{(k)} = \min\left\{d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}\right\}$

Time complexity: $\Theta(|V|^3)$, or $\Theta(n^3)$

Floyd-Warshall: Predecessor Matrix

- Recall that we also need to compute a ***predecessor matrix*** $\Pi = (\pi_{i,j})$

$$\pi_{i,j} = \begin{cases} \text{Nil} & i = j \text{ or no path from } i \text{ to } j \\ \text{Predecessor of } j \text{ on a shortest path from } i \text{ to } j & \text{otherwise} \end{cases}$$

  - i-th row of Π defines a shortest-paths tree rooted at i

Floyd-Warshall: Predecessor Matrix

- Recall that we also need to compute a ***predecessor matrix*** $\Pi = (\pi_{i,j})$

$$\pi_{i,j} = \begin{cases} \text{Nil} & i = j \text{ or no path from } i \text{ to } j \\ \text{Predecessor of } j \text{ on a shortest path from } i \text{ to } j & \text{otherwise} \end{cases}$$

  - i-th row of Π defines a shortest-paths tree rooted at i

- We have $\Pi^{(k)} = \left(\pi_{i,j}^{(k)}\right)$ corresponding to each $D^{(k)}$

$$\pi_{i,j}^{(k)} = \begin{cases} \text{Nil} & \begin{array}{l} i = j \text{ or no path from } i \text{ to } j \\ \text{with intermediate vertices } \leq k \end{array} \\ \begin{array}{l} \text{Predecessor of } j \text{ on a shortest path from } i \text{ to } j \\ \text{with intermediate vertices } \leq k \end{array} & \text{otherwise} \end{cases}$$

- We simply let $\Pi = \Pi^{(n)}$

Floyd-Warshall: Predecessor Matrix

- Base case

$$\pi_{i,j}^{(0)} = \begin{cases} \text{Nil} & \text{if } i = j \text{ or } (i,j) \notin E \\ i & \text{if } i \neq j \text{ and } (i,j) \in E \end{cases}$$

Floyd-Warshall: Predecessor Matrix

- Base case

$$\pi_{i,j}^{(0)} = \begin{cases} \text{Nil} & \text{if } i = j \text{ or } (i,j) \notin E \\ i & \text{if } i \neq j \text{ and } (i,j) \in E \end{cases}$$

- General case

$$\pi_{i,j}^{(k)} = \begin{cases} & \text{if } d_{i,j}^{(k-1)} \leq d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \\ & \text{if } d_{i,j}^{(k-1)} > d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \end{cases}$$

Floyd-Warshall: Predecessor Matrix

- Base case

$$\pi_{i,j}^{(0)} = \begin{cases} \text{Nil} & \text{if } i = j \text{ or } (i,j) \notin E \\ i & \text{if } i \neq j \text{ and } (i,j) \in E \end{cases}$$

- General case

$$\pi_{i,j}^{(k)} = \begin{cases} \pi_{i,j}^{(k-1)} & \text{if } d_{i,j}^{(k-1)} \leq d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \\ \pi_{k,j}^{(k-1)} & \text{if } d_{i,j}^{(k-1)} > d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \end{cases}$$

Floyd-Warshall: With Π matrix update

Floyd-Warshall(W)

```
Initialize D^(0) and Π^(0)
for k = 1, ..., n
    for i = 1, ..., n
        for j = 1, ..., n
            if d_{i,j}^(k-1) ≤ d_{i,k}^(k-1) + d_{k,j}^(k-1)
                d_{i,j}^(k) = d_{i,j}^(k-1)
                π_{i,j}^(k) = π_{i,j}^(k-1)
            else
                d_{i,j}^(k) = d_{i,k}^(k-1) + d_{k,j}^(k-1)
                π_{i,j}^(k) = π_{k,j}^(k-1)
```