Introduction to Algorithms and Data Structures

Lecture 19: All-pairs shortest paths via Dynamic Programming

Mary Cryan

School of Informatics
University of Edinburgh

“All pairs” shortest paths in graphs

We return to the world of graphs and directed graphs (following Lects 14-16).

In this lecture we again consider *weighted* graphs (and digraphs) $G = (V, E)$ where there is a weight function $w : E \rightarrow \mathbb{R}$ defining weights for all arcs/edges.

We are interested in evaluating the cost of shortest paths (from specific node u to specific node v) in the given weighted graph.

> *We will focus on all pairs shortest paths where we want to find the value of the minimum-cost path from u to v, for* every $u, v \in V$.

"All pairs" shortest paths in graphs

We return to the world of graphs and directed graphs (following Lects 14-16).

In this lecture we again consider *weighted* graphs (and digraphs) $G = (V, E)$ where there is a weight function $w : E \to \mathbb{R}$ defining weights for all arcs/edges.

We are interested in evaluating the cost of shortest paths (from specific node u to specific node v) in the given weighted graph.

> *We will focus on all pairs shortest paths where we want to find the value of the minimum-cost path from u to v, for* every $u, v \in V$.

(we will allow negative weights, but will assume there is no cycle with total negative weight)

Very different approach from Dijkstra's Algorithm!

All-pairs shortest paths

Given: A graph or digraph $G = (V, E)$ together with a weight function $w : E \rightarrow \mathbb{R}$ on edges. (assume V is in 1-1 correspondence with $\{0, \ldots, n-1\}$).

Problem: Compute a matrix $D \in \mathbb{R}^{n \times n}$ such that $D[i, j]$ is the value of the shortest-path (wrt w) from i to j, for every $0 \leq i, j \leq n-1$.

(possibly useful in the backend for a routefinding app)

Definition

Let $u, v \in V$ and suppose we have a path $p = e_1, \ldots, e_{|p|}$ from u to v (u is the source of e_1 and v the destination of $e_{|p|}$. Then the cost d_p *of this path* is

$$\sum_{h=1}^{|p|} w(e_h).$$

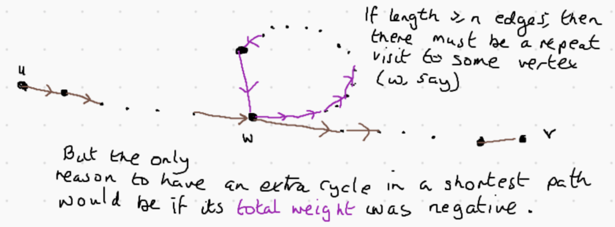
"Different $u \rightarrow v$ paths have different costs, we want the path with minimum cost" (and we want this for every u, v)

All-pairs shortest paths (preliminaries)

Lemma

Let $i, j \in V$, and suppose that there is at least one path from u to v in $G = (V, E)$.
Assume there is no cycle of total negative weight in G, w.
Then the minimum-cost path from u to v has length $\leq n - 1$.

Argument holds by contradiction



Also tells us we'll only see any particular vertex w once inside a S.P.

Floyd-Warshall: the idea

The Floyd-Warshall algorithm is defined around a trick where we consider "paths through low-index vertices" ... and increase the pool of "low-index vertices" by 1 each time.

Let $V_k = \{0, \ldots, k-1\}$ for every $1 \leq k \leq n$.

Let $\mathcal{P}_k = \{p : \text{p a path in } G \text{ with internal vertices restricted to } V_k\}$.

(endpoints of the paths can have higher indices)

Floyd-Warshall: the idea

Want to think about how we can compute
the min-cost-path(i,j) for every $i,j \in V$.

The smaller subproblems?

The min-cost-path values that restrict the
intermediate vertices of candidate paths
to $V_k = \{0, \ldots, k-1\}$

And as we work ... we will increase k.
- start with k=0 (V_0 is empty)
- next with k=1 ($V_1 = \{0\}$)
- on to k=2 ($V_2 = \{0,1\}$)

....

And so on, until k is n.

Floyd-Warshall: the idea

Remember $V_k = \{0, \dots, k-1\}$

And P_k the set of paths with interior restricted to V_k

$\boxed{D^{<k} = \begin{array}{l} n \times n \text{ matrix with} \\ \text{answers for } V_k,\ P_k \end{array}}$

use this to make D^{k+1}

$V_{k+1} = \{0, \dots, k\}$

path will use k <u>at most once</u>

TWO CASES:

- k is not used: $D^{<k}[i,j]$
- k is used <u>once</u>: $D^{<k}[i,k] + D^{<k}[k,j]$

TAKE MIN

All-pairs shortest paths (Floyd-Warshall)

Definition

For $0 \leq i, j \leq n-1$ and $k \geq 0$, let

$$d_{ij}^{<k} = \begin{cases} 0 & i = j \\ \min\{d_p : p \in \mathcal{P}_k, p \text{ is from } i \text{ to } j\} & i \neq j,\ i \to j \text{ paths exist in } \mathcal{P}_k \\ \infty & i \neq j, \text{ no } \mathcal{P}_k \text{ path for } i \to j. \end{cases}$$

Let $D^{<k} = (d_{ij}^{<k})_{0 \leq i,j \leq n-1}$; we may call $D^{<k}$ the *distance up to k* matrix of G.

Revision: A better description of $D^{<k}$ is to say it is the matrix of *distances via V_k-restricted paths*. After all, it is not the (distances) that are required to be low, only the indices of the vertices allowed to appear along a candidate path.

All-pairs shortest paths (Floyd-Warshall) - recurrence

Let $k \geq 0$ and consider the minimum-cost path for $i \to j$ in $\mathcal{P}_{k+1}$ (for any $0 \leq i, j \leq n-1$), if such a path exists. Two cases:

(a) The vertex k itself lies inside this minimum $i \to j$ path of $\mathcal{P}_{k+1}$
(*but assuming no negative cycles, will only appear once*)
Then $D^{<k+1}[i,j] = D^{<k}[i,k] + D^{<k}[k,j]$.

(b) Vertex k is **not** in the interior of the minimum $i \to j$ path of $\mathcal{P}_{k+1}$.
Then $D^{<k+1}[i,j] = D^{<k}[i,j]$.

Recurrence:

$$D^{<k+1}[i,j] = \begin{cases} 0 & i = j \\ \min\{D^{<k}[i,j], D^{<k}[i,k] + D^{<k}[k,j]\} & \text{otherwise} \end{cases}$$

Base case:

$$D^{<0}[i,j] = \begin{cases} 0 & i = j \\ w(i,j) & \text{if } i \neq j,\ (i,j) \in E \\ \infty & i \neq j, (i,j) \notin E \end{cases}$$

We let $d + \infty = \infty + d = \infty + \infty = \infty$ for all integers $d \geq 0$.

All-pairs shortest paths (Floyd-Warshall) - Algorithm

Algorithm FloydWarshall(G, w)

```
Initialise D^{<0} using Base case details
for k = 0 to n − 1 do
    for i = 0 to n − 1 do
        for j = 0 to n − 1 do
            D^{<k+1}[i, j] ← D^{<k}[i, j]        //Default option
            if j ≠ i and (D^{<k}[i, k] + D^{<k}[k, j]) < D^{<k+1}[i, j]
                D^{<k+1}[i, j] ← D^{<k}[i, k] + D^{<k}[k, j]
return D^{<n}
```

In practice we don't need all these arrays: we just need D^{curr} and D^{next}, and we can re-use ...

Getting the actual paths (Floyd-Warshall)

Build "predecessor" arrays $\Pi^{<k}$ in partnership with the $D^{<k}$ arrays.
$\Pi^{<k}[i,j]$ is the index of the vertex that appears *directly before* j in the shortest path from i to j subject to the restriction of intermediate vertices to V_k.

Body of the loop changes to:

5. $D^{<k+1}[i,j] \leftarrow D^{<k}[i,j]$ //Default option
   $\Pi^{<k+1}[i,j] \leftarrow \Pi^{<k}[i,j]$ //Copy "predecessor" (of j) too
6. **if** $j \neq i$ **and** $(D^{<k}[i,k] + D^{<k}[k,j]) < D^{<k+1}[i,j]$
7. $\quad D^{<k+1}[i,j] \leftarrow D^{<k}[i,k] + D^{<k}[k,j]$
   $\quad \Pi^{<k+1}[i,j] \leftarrow \Pi^{<k}[k,j]$
   $\quad$ //"Predecessor" (of j) is from the $D^{<k}[k,j]$ subpath

Then we can carry out a recursive "trace-back" starting from $\Pi^{<n}[i,j]$ to build the path achieving shortest path value $D^{<n}[i,j]$, for any i,j.

Running time is *Theta*(n^3) time

line 1. The matrix $D^{<0}$ is essentially (almost) the weights matrix of the graph. It's computable in $O(n^2)$ time ... just copy over.

lines 2.-7. The "triple loop"

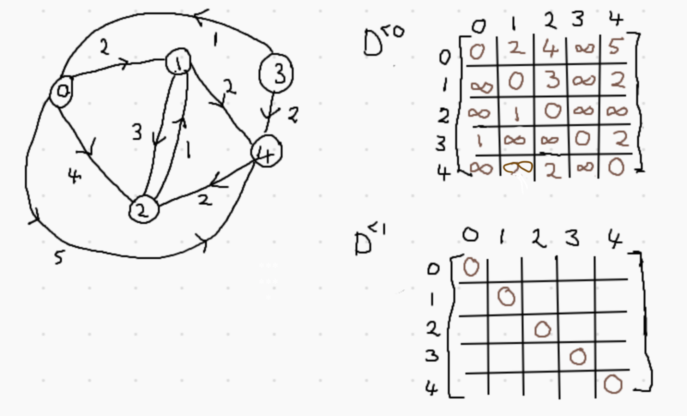
- We have n iterations of the outer loop with k
- n iterations of the middle loop with i
- n iterations of the inner loop with j
- ... and the body (lines 5.-7.) are $O(1)$

Hence we have $n \cdot n \cdot n \cdot O(1)$, giving $O(n^3)$.

$\Omega(n^3)$ Easier than usual to see the matching $\Omega(n^3)$ bound.

- The iterations of medium/inner loop are independent of the loops outside, hence we also have $n \cdot n \cdot n \cdot \Omega(1)$.

Homework example



Follow the Algorithm to build $D^{<1}, D^{<2}, D^{<3}, D^{<4}, D^{<5}$
(and also the Π arrays, if you like)

Reading

Even though this is named the "Floyd-Warshall" Algorithm (from around 1962), (essentially) the same Algorithm had previously been published by the French mathematician Bernard Roy in 1959.

Reading:

- For the Floyd-Warshall Algorithm, the relevant sections of [CLRS] are Sections 25.1 and 25.2.
- Some of the content in Chapter 24 of [CLRS] (about Single Source) is helpful.