All-Pairs Shortest Paths

The Floyd-Warshall Algorithm

The Problem

Given a directed graph $G = (V, E)$ with edge weights $w(u, v)$, compute $\text{dist}(i, j)$ for every pair of vertices i, j.

The output is an $n \times n$ matrix D with $D[i][j] = \text{dist}(i, j)$.

The Problem

Given a directed graph $G = (V, E)$ with edge weights $w(u, v)$, compute $\text{dist}(i, j)$ for every pair of vertices i, j.

The output is an $n \times n$ matrix D with $D[i][j] = \text{dist}(i, j)$.

First idea: run a single-source algorithm from every vertex.

weights	run n times	time	dense graph ($m \approx n^2$)
unweighted	BFS	$O(n(n+m))$	$O(n^3)$
≥ 0	Dijkstra	$O(n(n+m)\log n)$	$O(n^3 \log n)$
any	Bellman-Ford	$O(n^2 m)$	$O(n^4)$

The Problem

Given a directed graph $G = (V, E)$ with edge weights $w(u, v)$, compute $\text{dist}(i, j)$ for every pair of vertices i, j.

The output is an $n \times n$ matrix D with $D[i][j] = \text{dist}(i, j)$.

First idea: run a single-source algorithm from every vertex.

weights	run n times	time	dense graph ($m \approx n^2$)
unweighted	BFS	$O(n(n+m))$	$O(n^3)$
≥ 0	Dijkstra	$O(n(n+m)\log n)$	$O(n^3 \log n)$
any	Bellman-Ford	$O(n^2 m)$	$O(n^4)$

Floyd-Warshall: any weights (no negative cycle), $O(n^3)$ time, and about ten lines of code.

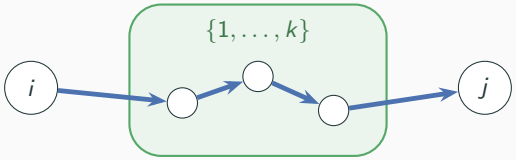
The Recurrence

Subproblems

Number the vertices $1, 2, \ldots, n$. The intermediate vertices of a path are all its vertices except the two endpoints.

Definition

$D^{(k)}[i][j]$ = length of a shortest path from i to j whose intermediate vertices all lie in $\{1, 2, \ldots, k\}$.



- $k = 0$: no intermediates, only direct edges.
- $k = n$: every vertex allowed, so $D^{(n)}[i][j] = \text{dist}(i, j)$.

Base Case

With no intermediate vertices, a path from i to j is either empty ($i = j$) or a single edge:

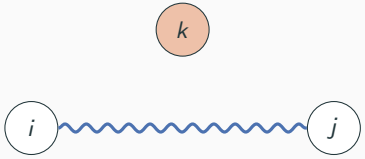
$$D^{(0)}[i][j] = \begin{cases} 0 & \text{if } i = j, \\ w(i,j) & \text{if } (i,j) \in E, \\ \infty & \text{otherwise.} \end{cases}$$

This is just the weighted adjacency matrix with zeros on the diagonal.

The Recurrence

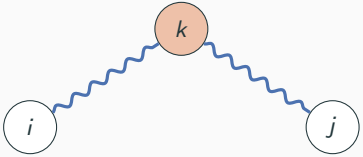
Now allow vertex k as well. A shortest path from i to j with intermediates in $\{1, \ldots, k\}$ either

avoids k



length $D^{(k-1)}[i][j]$

goes through k

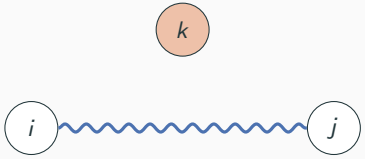


length $D^{(k-1)}[i][k] + D^{(k-1)}[k][j]$

The Recurrence

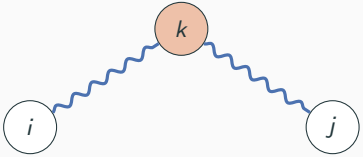
Now allow vertex k as well. A shortest path from i to j with intermediates in $\{1, \ldots, k\}$ either

avoids k



length $D^{(k-1)}[i][j]$

goes through k



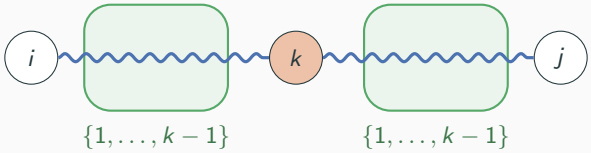
length $D^{(k-1)}[i][k] + D^{(k-1)}[k][j]$

Recurrence

$$D^{(k)}[i][j] = \min\big(D^{(k-1)}[i][j],\ D^{(k-1)}[i][k] + D^{(k-1)}[k][j]\big)$$

Why the Two Halves Use Only $\{1, \ldots, k-1\}$

Suppose the path goes through k. With no negative cycle, we may assume it is simple, so k appears exactly once.



- The part $i \rightsquigarrow k$ has k as an endpoint, not an intermediate. The same holds for $k \rightsquigarrow j$.
- Each part must itself be shortest among such paths, so they have lengths $D^{(k-1)}[i][k]$ and $D^{(k-1)}[k][j]$.

The Algorithm

Floyd-Warshall

```
function floyd_warshall(n, W)
    D = copy(W) # D^(0)
    for k in 1:n
        for i in 1:n
            for j in 1:n
                if D[i,k] + D[k,j] < D[i,j]
                    D[i,j] = D[i,k] + D[k,j]
                end
            end
        end
    end
    return D
end
```

- The outer loop must be over k: $D^{(k)}$ needs all of $D^{(k-1)}$.
- One matrix, updated in place, instead of $n+1$ matrices.
- Time $O(n^3)$, space $O(n^2)$.

Why Updating in Place Is Safe

In round k we read $D[i][k]$ and $D[k][j]$ while possibly overwriting other entries. Do these two entries change during round k?

Why Updating in Place Is Safe

In round k we read $D[i][k]$ and $D[k][j]$ while possibly overwriting other entries. Do these two entries change during round k?

- The update for entry $[i][k]$ is

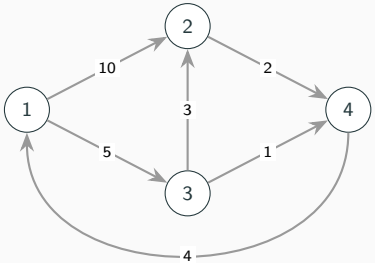
$$D[i][k] \leftarrow \min\big(D[i][k],\ D[i][k] + D[k][k]\big).$$

- $D[k][k] = 0$ unless there is a negative cycle through k.
- So $D[i][k]$ stays the same, and likewise $D[k][j]$.

Row k and column k are fixed during round k, so it does not matter whether we read old or new values elsewhere.

Example

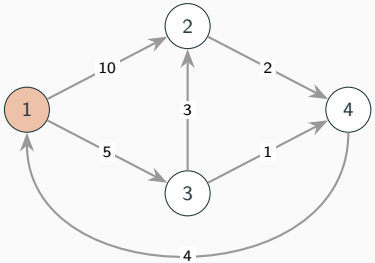
Example: $D^{(0)}$



	1	2	3	4
1	0	10	5	∞
2	∞	0	∞	2
3	∞	3	0	1
4	4	∞	∞	0

Direct edges only.

Example: $D^{(1)}$, through vertex 1

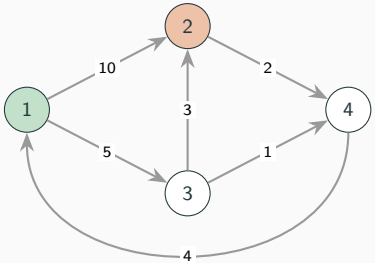


	1	2	3	4
1	0	10	5	∞
2	∞	0	∞	2
3	∞	3	0	1
4	4	**14**	**9**	0

Only vertex 4 has an edge into 1.

$D[4][2] = 4 + 10 = 14, \quad D[4][3] = 4 + 5 = 9.$

Example: $D^{(2)}$, through vertices 1, 2



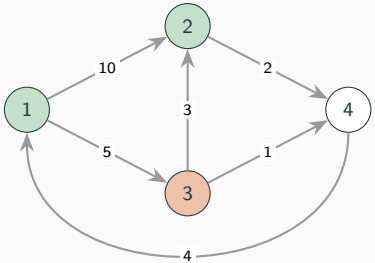
	1	2	3	4
1	0	10	5	**12**
2	∞	0	∞	2
3	∞	3	0	1
4	4	14	9	0

Vertex 2 only leads to 4.

$D[1][4] = 10 + 2 = 12$.

($D[3][4]$: $3 + 2 = 5$ does not beat 1.)

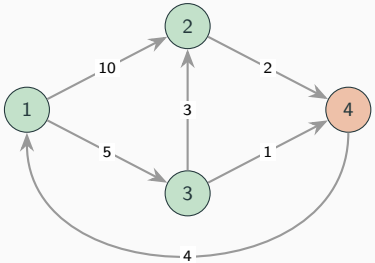
Example: $D^{(3)}$, through vertices 1, 2, 3



	1	2	3	4
1	0	**8**	5	**6**
2	∞	0	∞	2
3	∞	3	0	1
4	4	**12**	9	0

$D[1][2] = 5 + 3 = 8, \quad D[1][4] = 5 + 1 = 6,$
$D[4][2] = 9 + 3 = 12.$

Example: $D^{(4)}$, every vertex allowed



	1	2	3	4
1	0	8	5	6
2	**6**	0	**11**	2
3	**5**	3	0	1
4	4	12	9	0

The edge $4 \to 1$ finally lets 2 and 3 reach 1.
$D[2][1] = 2 + 4 = 6, \quad D[2][3] = 2 + 9 = 11,$
$D[3][1] = 1 + 4 = 5.$

Correctness and Extensions

Correctness

Claim

After round k, $D[i][j] = D^{(k)}[i][j]$ for all i, j.

Proof by induction on k.

- $k = 0$: D starts as $D^{(0)}$ by construction.
- $k \geq 1$: assume $D = D^{(k-1)}$ before round k. Row k and column k do not change during the round, so each update computes

$$\min\big(D^{(k-1)}[i][j],\ D^{(k-1)}[i][k] + D^{(k-1)}[k][j]\big),$$

  which is $D^{(k)}[i][j]$ by the recurrence.

With $k = n$ every vertex may be an intermediate, so $D[i][j] = \mathrm{dist}(i, j)$. □

Recovering the Paths

Keep a second matrix $P[i][j]$ = the vertex just before j on the best path from i to j found so far.

- Start: $P[i][j] = i$ if $(i, j) \in E$, otherwise undefined.
- When the path through k wins: $P[i][j] \leftarrow P[k][j]$, since the path now ends the same way as the best path from k to j.

Recovering the Paths

Keep a second matrix $P[i][j] =$ the vertex just before j on the best path from i to j found so far.

- Start: $P[i][j] = i$ if $(i, j) \in E$, otherwise undefined.
- When the path through k wins: $P[i][j] \leftarrow P[k][j]$, since the path now ends the same way as the best path from k to j.

To print the path from i to j, follow P backwards from j:

$$j,\ P[i][j],\ P[i][P[i][j]],\ \ldots,\ i.$$

Recovering the Paths

Keep a second matrix $P[i][j]$ = the vertex just before j on the best path from i to j found so far.

- Start: $P[i][j] = i$ if $(i, j) \in E$, otherwise undefined.
- When the path through k wins: $P[i][j] \leftarrow P[k][j]$, since the path now ends the same way as the best path from k to j.

To print the path from i to j, follow P backwards from j:

$$j,\ P[i][j],\ P[i][P[i][j]],\ \ldots,\ i.$$

Example ($4 \to 2$, length 12): $P[4][2] = 3$, $P[4][3] = 1$, $P[4][1] = 4$.
So the path is $4 \to 1 \to 3 \to 2$, with length $4 + 5 + 3 = 12$.

Detecting Negative Cycles

Floyd-Warshall assumes there is no negative cycle. If there is one, it shows up on the diagonal.

- Every value $D[i][i]$ is the length of some closed walk through i, so $D[i][i] < 0$ means there is a negative cycle.
- Conversely, let C be a negative cycle, k its largest vertex and $i \neq k$ another vertex of C. The two arcs of C between i and k use only intermediates below k, so after round k, $D[i][i] \leq \text{length}(C) < 0$.

Check

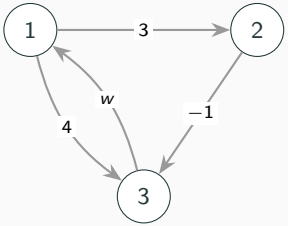
After the algorithm finishes: there is a negative cycle $\iff$ $D[i][i] < 0$ for some i.

Floyd-Warshall in Julia

```
# W[i,j] = edge weight, Inf if no edge, 0 on the diagonal
function floyd_warshall(W)
    n = size(W, 1)
    D = copy(W)
    for k in 1:n
        for i in 1:n
            D[i,k] == Inf && continue # nothing to gain
            for j in 1:n
                if D[i,k] + D[k,j] < D[i,j]
                    D[i,j] = D[i,k] + D[k,j]
                end
            end
        end
    end
    for i in 1:n
        D[i,i] < 0 && error("negative cycle detected")
    end
    return D
end
```

- `D` is an $n \times n$ matrix, indexed `D[i,j]`.
- Skipping rows with `D[i,k] == Inf` avoids pointless work on sparse graphs.

Exercise



1. Take $w = 2$. Write down $D^{(0)}$, then run Floyd-Warshall and give $D^{(1)}, D^{(2)}, D^{(3)}$.
2. Now take $w = -3$. What does the algorithm report, and why?

Solution

(a) $w = 2$:

$D^{(0)}$	1	2	3
1	0	3	4
2	∞	0	-1
3	2	∞	0

$D^{(1)}$	1	2	3
1	0	3	4
2	∞	0	-1
3	2	**5**	0

$D^{(2)}$	1	2	3
1	0	3	**2**
2	∞	0	-1
3	2	5	0

$D^{(3)}$	1	2	3
1	0	3	2
2	**1**	0	-1
3	2	5	0

For example $D[1][3] = 3 + (-1) = 2$ goes through 2, beating the direct edge of weight 4, and $D[2][1] = -1 + 2 = 1$ goes through 3.

Solution

(a) $w = 2$:

$D^{(0)}$	1	2	3
1	0	3	4
2	∞	0	-1
3	2	∞	0

$D^{(1)}$	1	2	3
1	0	3	4
2	∞	0	-1
3	2	**5**	0

$D^{(2)}$	1	2	3
1	0	3	**2**
2	∞	0	-1
3	2	5	0

$D^{(3)}$	1	2	3
1	0	3	2
2	**1**	0	-1
3	2	5	0

For example $D[1][3] = 3 + (-1) = 2$ goes through 2, beating the direct edge of weight 4, and $D[2][1] = -1 + 2 = 1$ goes through 3.

(b) $w = -3$: the cycle $1 \to 2 \to 3 \to 1$ has length $3 - 1 - 3 = -1 < 0$. The final matrix has $D[1][1] = -1$ (and $D[2][2] = -1$, $D[3][3] = -2$), so the algorithm reports a negative cycle. The other entries are meaningless, since no shortest paths exist.

Summary

- $D^{(k)}[i][j]$: shortest $i \to j$ path whose intermediates lie in $\{1, \ldots, k\}$.
- Recurrence: either avoid k, or go $i \rightsquigarrow k \rightsquigarrow j$.
- Three nested loops with k outermost, one matrix updated in place.
- $O(n^3)$ time, $O(n^2)$ space, negative edges allowed.
- A negative entry on the diagonal reveals a negative cycle.

situation	best choice	time
sparse, weights ≥ 0	Dijkstra from each vertex	$O(nm \log n)$
dense, or negative weights	Floyd-Warshall	$O(n^3)$