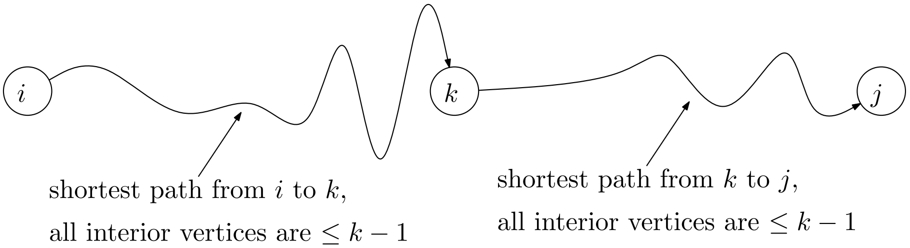
§5.5 All-Pairs Shortest Paths

Let $G = (V, E)$ be a directed graph, where $V = \{1, 2, ..., n\}$. Each edge (i, j) has a weight $wt(i, j) > 0$.

For all i and j, compute the weight of a shortest path in G from i to j, which we denote by $\delta_G(i, j)$.

Step 1: Structure of the Optimal Solution

Consider the shortest path from i to j, and assume this path has at least one interior vertex. Let k be the largest interior vertex.



Step 2: Set Up a recurrence for the Optimal Solution

For

$$1 \le i \le n \qquad 1 \le j \le n \qquad 0 \le k \le n,$$

let $dist(i, j, k)$ be the length of a shortest path from i to j, all of whose interior vertices are $\le k$.

Step 2: Set Up a recurrence for the Optimal Solution

For

$$1 \leq i \leq n \qquad 1 \leq j \leq n \qquad 0 \leq k \leq n,$$

let $dist(i, j, k)$ be the length of a shortest path from i to j, all of whose interior vertices are $\leq k$.

We want to compute

$$dist(i, j, n) = \delta_G(i, j)$$

for all $1 \leq i \leq n$, $1 \leq j \leq n$.

Recurrence:

- For $1 \leq i \leq n$, $dist(i, i, 0) = 0$.

Recurrence:

- For $1 \leq i \leq n$, $dist(i, i, 0) = 0$.
- For $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$,

$$dist(i, j, 0) = \begin{cases} wt(i, j) & \text{if } (i, j) \text{ is an edge,} \\ \infty & \text{otherwise.} \end{cases}$$

Recurrence:

- For $1 \leq i \leq n$, $dist(i,i,0) = 0$.
- For $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$,

$$dist(i,j,0) = \begin{cases} wt(i,j) & \text{if } (i,j) \text{ is an edge,} \\ \infty & \text{otherwise.} \end{cases}$$

- For $1 \leq i \leq n$, $1 \leq j \leq n$, $1 \leq k \leq n$,

$$dist(i,j,k) = \min\left\{dist(i,j,k-1), dist(i,k,k-1) + dist(k,j,k-1)\right\}.$$

Step 3: Solve the Recurrence Bottom-Up

Algorithm Floyd-Warshall

```
for i = 1 to n do
    for j = 1 to n do
        if i = j then
            dist(i, j, 0) = 0
        else
            dist(i, j, 0) = ∞
        end if
    end for
end for
for all edges (i, j) do
    dist(i, j, 0) = wt(i, j)
end for
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            dist(i, j, k) = min {dist(i, j, k − 1), dist(i, k, k − 1) + dist(k, j, k − 1)}
        end for
    end for
end for
```

Running time: $O(n^3)$