Lecture 14: All-Pairs Shortest Paths

Revised May 1, 2003

CLRS Section 25.1

Outline of this Lecture

- Introduction of the all-pairs shortest path problem.

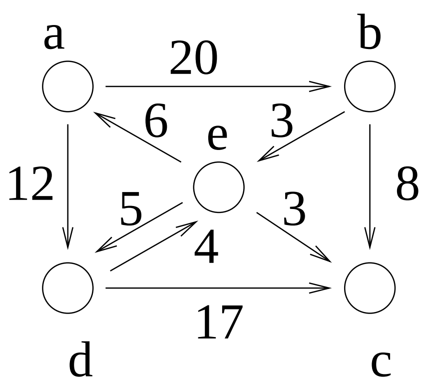
- First solution using Dijkstra's algorithm.
  Assumes no negative weight edges
  $\Theta\left(|V|^3 \log |V|\right)$.
  Needs priority queues

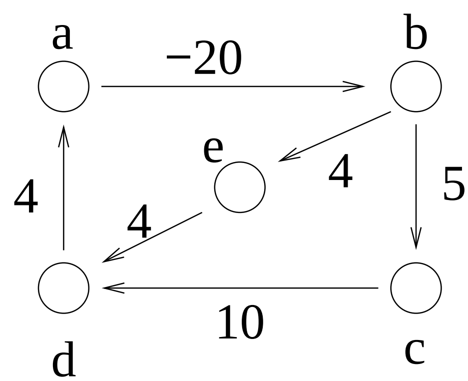
- A (first) dynamic programming solution.
  Only assumes no negative weight cycles.
  First version is $\Theta\left(|V|^4\right)$.
  Repeated squaring reduces to $\Theta\left(|V|^3 \log |V|\right)$.

  No special data structures needed.

The All-Pairs Shortest Paths Problem

Given a weighted digraph $G = (V, E)$ with weight function $w : E \rightarrow \mathbf{R}$, ($R$ is the set of real numbers), determine the length of the shortest path (i.e., distance) between all pairs of vertices in G. Here we assume that there are no cycles with zero or negative cost.





without negative cost cycle   with negative cost cycle

Solution 1: Using Dijkstra's Algorithm

If there are no negative cost edges apply Dijkstra's algorithm to each vertex (as the source) of the digraph.

- Recall that D's algorithm runs in $\Theta((n+e)\log n)$. This gives a

  $$\Theta(n(n+e)\log n) = \Theta(n^2 \log n + ne \log n)$$

  time algorithm, where $n = |V|$ and $e = |E|$.

- If the digraph is dense, this is an $\Theta(n^3 \log n)$algorithm.

- With more advanced (complicated) data structures D's algorithm runs in $\Theta(n \log n + e)$time yielding a $\Theta(n^2 \log n + ne)$ final algorithm. For dense graphs this is $\Theta(n^3)$ time.

Solution 2: Dynamic Programming

(1) How do we decompose the all-pairs shortest paths problem into subproblems

(2) How do we express the optimal solution of a subproblem in terms of optimal solutions to some subsubproblems?

(3) How do we use the recursive relation from (2) to compute the optimal solution in a bottom-up fashion?

(4) How do we construct all the shortest paths?

Solution 2: Input and Output Formats

To simplify the notation, we assume that $V = \{1, 2, \ldots, n\}$.

Assume that the graph is represented by an $n \times n$ matrix with the weights of the edges:

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

Output Format: an $n \times n$ matrix $D = [d_{ij}]$ where d_{ij} is the length of the shortest path from vertex i to j.

Step 1: How to Decompose the Original Problem

- Subproblems with smaller sizes should be easier to solve.
- An optimal solution to a subproblem should be expressed in terms of the optimal solutions to subproblems with smaller sizes.

These are guidelines ONLY.

Step 1: Decompose in a Natural Way

- Define $d_{ij}^{(m)}$ to be the length of the shortest path from i to j that contains at most m edges.
  Let $D^{(m)}$ be the $n \times n$ matrix $[d_{ij}^{(m)}]$.

- $d_{ij}^{(n-1)}$ is the true distance from i to j (see next page for a proof this conclusion).

- Subproblems: compute $D^{(m)}$ for $m = 1, \cdots, n - 1$.

Question: Which $D^{(m)}$ is easiest to compute?

$d_{ij}^{(n-1)}$ **= True Distance from i to j**

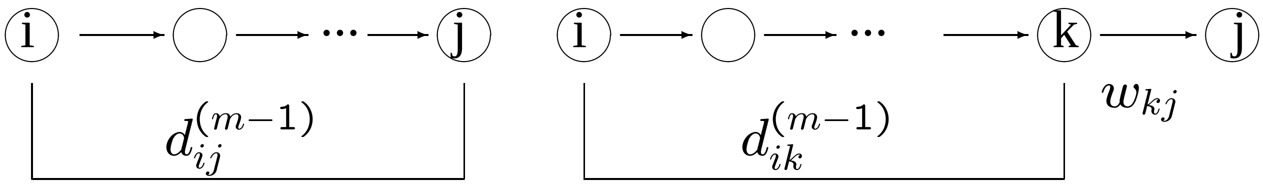
Proof: We prove that any shortest path P from i to j contains at most $n-1$ edges.

First note that since all cycles have positive weight, a shortest path can have no cycles (if there were a cycle, we could remove it and lower the length of the path).

A path without cycles can have length at most $n-1$ (since a longer path must contain some vertex twice, that is, contain a cycle).

A Recursive Formula

Consider a shortest path from i to j of length $d_{ij}^{(m)}$.



Case 1: at most $m-1$ edges shortest path

Case 2: exactly m edges shortest path

Case 1: It has at most $m-1$ edges.
Then $d_{ij}^{(m)} = d_{ij}^{(m-1)} = d_{ij}^{(m-1)} + w_{jj}$.
Case 2: It has m edges. Let k be the vertex before j on a shortest path.
Then $d_{ij}^{(m)} = d_{ik}^{(m-1)} + w_{kj}$.

Combining the two cases,

$$d_{ij}^{(m)} = \min_{1 \le k \le n} \left\{ d_{ik}^{(m-1)} + w_{kj} \right\}.$$

Step 3: Bottom-up Computation of $D^{(n-1)}$

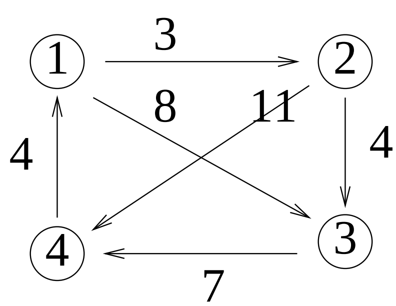
- Bottom: $D^{(1)} = \left[w_{ij} \right]$, the weight matrix.

- Compute $D^{(m)}$ from $D^{(m-1)}$, for $m = 2, ..., n-1$, using

$$d_{ij}^{(m)} = \min_{1 \le k \le n} \left\{ d_{ik}^{(m-1)} + w_{kj} \right\}.$$

Example: Bottom-up Computation of $D^{(n-1)}$

Example

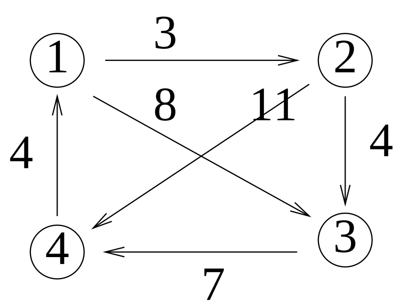


$D^{(1)} = [w_{ij}]$ is just the weight matrix:

$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty \\ \infty & 0 & 4 & 11 \\ \infty & \infty & 0 & 7 \\ 4 & \infty & \infty & 0 \end{bmatrix}$$

Example: Computing $D^{(2)}$ from $D^{(1)}$

$$d_{ij}^{(2)} = \min_{1 \le k \le 4} \left\{ d_{ik}^{(1)} + w_{kj} \right\}.$$

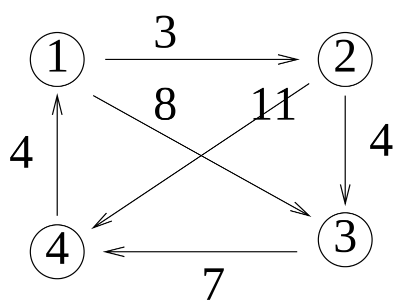


With $D^{(1)}$ given earlier and the recursive formula,

$$D^{(2)} = \begin{bmatrix} 0 & 3 & 7 & 14 \\ 15 & 0 & 4 & 11 \\ 11 & \infty & 0 & 7 \\ 4 & 7 & 12 & 0 \end{bmatrix}$$

Example: Computing $D^{(3)}$ from $D^{(2)}$

$$d_{ij}^{(3)} = \min_{1 \le k \le 4} \left\{ d_{ik}^{(2)} + w_{kj} \right\}$$



With $D^{(2)}$ given earlier and the recursive formula,

$$D^{(3)} = \begin{bmatrix} 0 & 3 & 7 & 14 \\ 15 & 0 & 4 & 11 \\ 11 & 14 & 0 & 7 \\ 4 & 7 & 11 & 0 \end{bmatrix}$$

$D^{(3)}$ gives the distances between any pair of vertices.

The Algorithm for Computing $D^{(n-1)}$

for $m = 1$ to $n - 1$
    for $i = 1$ to n
        for $j = 1$ to n
        $\{$
            $min = \infty$;
            for $k = 1$ to n
            $\{$
                $new = d_{ik}^{(m-1)} + w_{kj}$;
                if ($new < min$) $min = new$;
            $\}$
            $d_{ij}^{(m)} = min$;
        $\}$

Comments on Solution 2

- Algorithm uses $\Theta(n^3)$ space; how can this be reduced down to $\Theta(n^2)$?

- How can we extract the actual shortest paths from the solution?

- Running time $O(n^4)$, much worse than the solution using Dijkstra's algorithm. Can we improve this?

Repeated Squaring

Q: Suppose we are given a number x and asked to calculate x^{2^i}. How many multiplications are needed?

A: Only $(i-1)$! Calculate

$$x^2 = x \cdot x, \quad x^4 = x^2 \cdot x^2, \quad \ldots, \quad x^{2^i} = x^{2^{i-1}} \cdot x^{2^{i-1}}$$

We saw that all shortest paths have distance $< n$.
In particular, this implies that $D^{\left(2^{\lceil \log_2 n \rceil}\right)} = D^{(n-1)}$.

We can calculate $D^{\left(2^{\lceil \log_2 n \rceil}\right)}$ using "repeated squaring" to find

$$D^{(2)}, D^{(4)}, D^{(8)}, \ldots, D^{\left(2^{\lceil \log_2 n \rceil}\right)}$$

We use the recurrence relation:

- Bottom: $D^{(1)} = \left[w_{ij}\right]$, the weight matrix.

- For $s, t \geq 1$ compute $D^{(s+t)}$ using

$$d_{ij}^{(s+t)} = \min_{1 \leq k \leq n} \left\{ d_{ik}^{(s)} + d_{kj}^{(s)} \right\}.$$

For proof of this recurrence relation see textbook (very similar to recurrence relation we proved earlier this lecture).

Given this relation we can calculate $D^{(2^i)}$ from $D^{(2^{i-1})}$ in $O(n^3)$ time. We can therefore calculate all of

$$D^{(2)},\ D^{(4)}, D^{(8)}, \ldots,\ D^{\left(2^{\lceil \log_2 n \rceil}\right)} = D^{(n)}$$

in $O(n^3 \log n)$ time, improving our running time.