Data Structures – LECTURE 16

All shortest paths algorithms

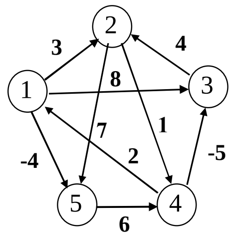
- Properties of all shortest paths
- Simple algorithm: $O(|V|^4)$ time
- Better algorithm: $O(|V|^3 \lg |V|)$ time
- Floyd-Warshall algorithm: $O(|V|^3)$ time

Chapter 25 in the textbook (pp 620–635).

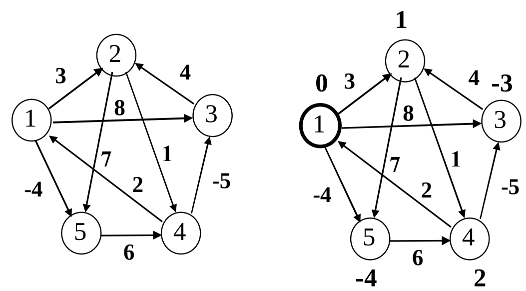
All shortest paths

- Generalization of the single source shortest-path problem.
- Simple solution: run the shortest path algorithm for each vertex → complexity is $O(|E|.|V|.|V|) = O(|V|^4)$ for Bellman-Ford and $O(|E|.\lg |V|.|V|) = O(|V|^3 \lg |V|)$ for Dijsktra.
- Can we do better? Intuitively it would seem so, since there is a lot of repeated work → exploit the optimal sub-path property.
- We indeed can do better: $O(|V|^3)$.

All-shortest paths: example (1)



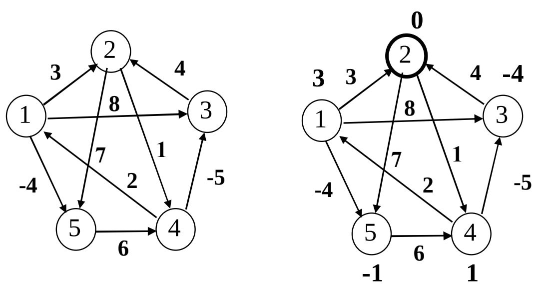
All-shortest paths: example (2)



dist (0 1 −3 2 −4)

pred (ϕ 3 4 5 1)

All-shortest paths: example (3)



dist (3 0 −4 1 −1)

pred (4 ϕ 4 2 1)

All shortest-paths: representation

We will use matrices to represent the graph, the shortest path lengths, and the predecessor sub-graphs.

- Edge matrix: entry (i,j) in adjacency matrix W is the weight of the edge between vertex i and vertex j.
- Shortest-path lengths matrix: entry (i,j) in L is the shortest path length between vertex i and vertex j.
- Predecessor matrix: entry (i,j) in Π is the predecessor of j on some shortest path from i (***null*** when $i = j$ or when there is no path).

All-shortest paths: definitions

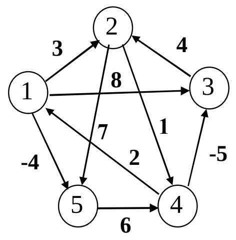
Edge matrix: entry (i,j) in adjacency matrix W is the weight of the edge between vertex i and vertex j.

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

Shortest-paths graph: the graphs $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$ are the shortest-path graphs rooted at vertex I, where:

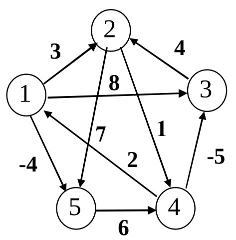
Example: edge matrix

$$W = \begin{array}{cc} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} & \\ \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \end{array}$$



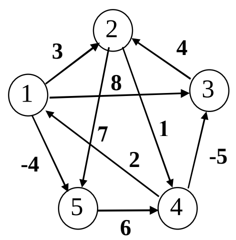
Example: shortest-paths matrix

$$L = \begin{array}{cc} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} & \\ \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \end{array}$$



Example: predecessor matrix

$$\Pi = \begin{array}{cc} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} & \\ \begin{pmatrix} null & 3 & 4 & 5 & 1 \\ 4 & null & 4 & 2 & 1 \\ 4 & 3 & null & 2 & 1 \\ 4 & 3 & 4 & null & 1 \\ 4 & 3 & 4 & 5 & null \end{pmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \end{array}$$



The structure of a shortest path

1. All sub-paths of a shortest path are shortest paths. Let $p = <v_1, .. v_k>$ be the shortest path from v_1 to v_k. The sub-path between v_i and v_j, where $1 \leq i,j \leq k$, $p_{ij} = <v_i, .. v_j>$ is a shortest path.
2. The shortest path from vertex i to vertex j with at most m edges is either:
   - the shortest path with at most (m-1) edges (no improvement)
   - the shortest path consisting of a shortest path within the (m-1) vertices + the weight of the edge from a vertex within the (m-1) vertices to an extra vertex m.

Recursive solution to all-shortest paths

Let $l^{(m)}_{ij}$ be the minimum weight of any path from vertex i to vertex j that has at most m edges.
When m=0:

$$l^{(0)}_{ij} = \begin{cases} 0 & i = j \\ \infty & i \neq j \end{cases}$$

For $m \geq 1$, $l^{(m)}_{ij}$ is the minimum of $l^{(m-1)}_{ij}$ and the shortest path going through the vertices neighbors:

All-shortest-paths: solution

- Let $W=(w_{ij})$ be the edge weight matrix and $L=(l_{ij})$ the all-shortest shortest path matrix computed so far, both $n\times n$.
- Compute a series of matrices $L^{(1)}, L^{(2)}, \ldots, L^{(n-1)}$ where for $m = 1,\ldots,n-1$, $L^{(m)} = (l^{(m)}_{ij})$ is the matrix with the all-shortest-path lengths with at most m edges. Initially, $L^{(1)} = W$, and $L^{(n-1)}$ containts the actual shortest-paths.
- Basic step: compute $L^{(m)}$ from $L^{(m-1)}$ and W.

Algorithm for extending all-shortest paths by one edge: from L$^{(m-1)}$ to L$^{(m)}$

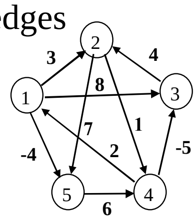
Extend-Shortest-Paths($L=(l_{ij}),W$)
$n \leftarrow rows[L]$
Let $L' =(l'_{ij})$ be an $n\times n$ *matrix.*
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to n **do**
    $l'_{ij} \leftarrow \infty$
    for $k \leftarrow 1$ to n **do**
      $l'_{ij} \leftarrow \min(l'_{ij}, l_{ik} + w_{kj})$
return L'

Complexity: $\Theta(|V|^3)$

This is exactly as matrix multiplication!

Matrix-Multiply(A,B)
$n \leftarrow rows[A]$
Let $C =(c_{ij})$ be an $n\times n$ *matrix.*
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to n **do**
    $c_{ij} \leftarrow 0$ $(l'_{ij} \leftarrow \infty)$
    for $k \leftarrow 1$ to n **do**
      $c_{ij} \leftarrow c_{ij}+ a_{ik}.b_{kj}$ $(l'_{ij} \leftarrow \min(l'_{ij}, l_{ij} + w_{kj}))$
return L'

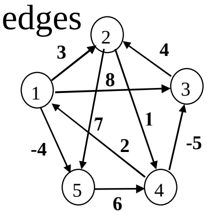
Paths with at most two edges



$$L^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

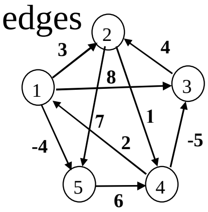
Paths with at most three edges



$$L^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

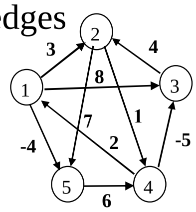
Paths with at most four edges



$$L^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Paths with at most five edges



$$L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$L^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

All-shortest paths algorithm

All-Shortest-Paths(W)
$n \leftarrow rows[W]$
$L^{(1)} \leftarrow W$
for $m \leftarrow 2$ to $n-1$ **do**
  $L^{(m)} \leftarrow$ Extend-Shortest-Paths($L^{(m-1)}$,W)
return $L^{(m)}$

Complexity: $\Theta(|V|^4)$

Improved all-shortest paths algorithm

- The goal is to compute the final $L^{(n-1)}$, not all the $L^{(m)}$
- We can avoid computing most $L^{(m)}$ as follows:

  $L^{(1)} = W$
  $L^{(2)} = W.W$ repeated squaring
  $L^{(4)} = W^4 = W^2.W^2$
  …

Since $2^{\lceil \lg(n-1) \rceil} \ge n-1$ the final product is equal to $L^{(n-1)}$
only $|\lg(n-1)|$ iterations are necessary!

Faster-all-shortest paths algorithm

Faster-All-Shortest-Paths(W)
$n \leftarrow rows[W]$
$L^{(1)} \leftarrow W$
while $m < n-1$ **do**
  $L^{(2m)} \leftarrow$ Extend-Shortest-Paths($L^{(m)}$,$L^{(m)}$)
  $m \leftarrow 2m$
return $L^{(m)}$

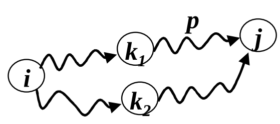
Complexity: $\Theta(|V|^3 \lg (|V|))$

Floyd-Warshall algorithm

- Assumes there are no negative-weight cycles.
- Uses a different characterization of the structure of the shortest path. It exploits the properties of the intermediate vertices of the shortest path.
- Runs in $O(|V|^3)$.

Structure of the shortest path (1)

- An intermediate vertex v_i of a simple path $p=<v_1,..,v_k>$ is any vertex other than v_1 or v_k.
- Let $V=\{1,2,\ldots,n\}$ and let $K=\{1,2,\ldots,k\}$ be a subset for $k \le n$. For any pair of vertices i,j in V, consider all paths from i to j whose intermediate vertices are drawn from K. Let p be the minimum-weight path among them.



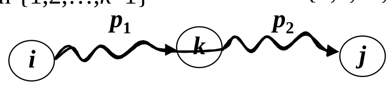
Structure of the shortest path (2)

1. <u>k is not an intermediate vertex of path p</u>:
   All vertices of path p are in the set $\{1,2,\ldots,k-1\}$
   → a shortest path from i to j with all intermediate vertices in $\{1,2,\ldots,k-1\}$ is also a shortest path with all intermediate vertices in $\{1,2,\ldots,k\}$.
2. <u>k is an intermediate vertex of path p</u>:
   Break p into two pieces: p_1 from i to k and p_2 from k to j. Path p_1 is a shortest path from i to k and path p_2 is a shortest path from k to j with all intermediate vertices in $\{1,2,\ldots,k-1\}$.

Structure of the shortest path (3)

all intermediate vertices in $\{1,2,\ldots,k-1\}$

all intermediate vertices in $\{1,2,\ldots,k-1\}$



all intermediate vertices in $\{1,2,\ldots,k\}$

Recursive definition

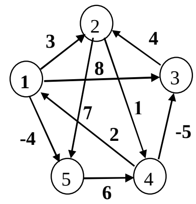
Let $d^{(k)}_{ij}$ be the weight of a shortest path from vertex i to vertex j for which all intermediate vertices are in the set $\{1,2,\ldots,k\}$. Then:

The matrices $D^{(k)} = (d^{(k)}_{ij})$ will keep the intermediate solutions.

Example: Floyd-Warshall algorithm (1)

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$



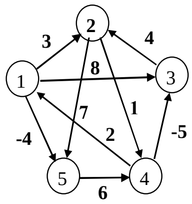
$K=\{1\}$

Improvements: (4,2) and (4,5)

Example: Floyd-Warshall algorithm (2)

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$



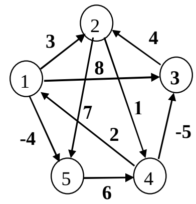
$K=\{1,2\}$

Improvements: (1,4) (3,4),(3,5)

Example: Floyd-Warshall algorithm (3)

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$



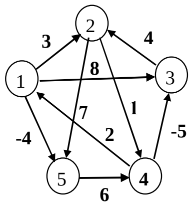
$K=\{1,2,3\}$

Improvement: (4,2)

Example: Floyd-Warshall algorithm (4)

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$



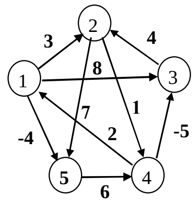
K={1,2,3,4}

Improvements: (1,3),(1,4)
(2,1), (2,3), (2,5)
(3,1),(3,5), (5,1),(5,2),(5,3)

Example: Floyd-Warshall algorithm (5)

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$



K={1,2,3,4,5}

Improvements:
(1,2),(1,3),(1,4)

Transitive closure (1)

- Given a directed graph G=(V,E) with vertices V = {1,2,…,n} determine for every pair of vertices (i,j) if there is a path between them.
- The *transitive closure graph* of G, G^*=(V,E^*) is such that E^* = {(i,j): if there is a path i and j}.
- Represent E^* as a binary matrix and perform logical binary operations AND (/\) and OR (\/) instead of *min* and + in the Floyd-Warshall algorithm.

Transitive closure (2)

The definition of the transitive closure is:

The matrices $T^{(k)}$ indicate if there is a path with at most k edges between s and i.

Transitive closure algorithm

Transitive-Closure(W)
n ← *rows*[W]
$T^{(0)}$ ← Binarize(W)
for k ← 1 to n **do**
  for i ← 1 to n **do**
    for j ← 1 to n **do**

return $T^{(n)}$

Complexity: Θ(|V|^3)

Summary

- Adjacency matrix representation is the most convenient for representing all-shortest-paths.
- Computing all shortest-paths is akin to taking the transitive closure of the edge weights.
- Matrix multiplication algorithm runs in $O(|V|^3 \lg |V|)$.
- The Floyd-Warshall algorithm improves paths through intermediate vertices instead of working on individual edges.
- Its running time: $O(|V|^3)$.

Other graph algorithms

- Many more interesting problems, including network flow, graph isomorphism, coloring, partition, etc.
- Problems can be classified by the type of solution.
- Easy problems: polynomial-time solutions $O(f(n))$ where $f(n)$ is a polynomial function of degree at most k.
- Hard problems: exponential-time solutions $O(f(n))$ where $f(n)$ is an exponential function, usually 2^n.

Easy graph problems

- Network flow – maximum flow problem
- Maximum bipartite matching
- Planarity testing and plane embedding.

Hard graph problems

- Graph and sub-graph isomorphism.
- Largest clique, Independent set
- Vertex tour (Traveling Salesman problem)
- Graph partition
- Vertex coloring

However, not all is lost!

- Good heuristics that perform well in most cases
- Polynomial-time approximation algorithms