csce750 — Analysis of Algorithms
Fall 2019 — Lecture Notes: All-Pairs Shortest Paths

This document contains slides from the lecture, formatted to be suitable for printing or individual reading, and with some supplemental explanations added. It is intended as a supplement to, rather than a replacement for, the lectures themselves — you should not expect the notes to be self-contained or complete on their own.

1 Introduction

CLRS 25

All-pairs shortest path problem:

- Input: A weighted directed graph G with no negative weights, stored as a weight matrix.
- Output: For each vertex pair v, w in G, a path in G from v to w that minimizes the total weight of edges crossed.

Specifically, we want:

- A **distance matrix** D, in which element d_{ij} represents the length of the shortest path from i to j.
- A **predecessor matrix** Π, in which element π_{ij} represents the predecessor of of j on some shortest path from i.

2 A simple option

We can solve APSP by solving each of the V single-source shortest path instances separately.

- Dijkstra with Fibonacci heap: $O(V^2 \lg V + VE)$

3 Can we do better?

Consider a path from v_I to v_G:

$$v_I \to v_1 \to v_2 \to \cdots v_m \to v_G$$

Observation: Finding the shortest path is essentially the problem of choosing which nodes will appear as **intermediate nodes** on the path.

4 Family of subproblems

We can form a family of subproblems by *restricting which nodes may appear as intermediate nodes*.

Define $d_{ij}^{(k)}$ as the length of the shortest path from i to j, using only the first k nodes (as they are ordered in the input weight matrix W) as possible intermediate nodes.

Note that when $k = n$, we get $d_{ij}^{(n)} = d_{ij}$.

5 Optimal substructure?

How can we find $d_{ij}^{(k)}$?

Two choices:

- Option 1:

  Do not include k as an intermediate node. Then $d_{ij}^{(k)} = d_{ij}^{(k-1)}$.

- Option 2:

  Include k as an intermediate node. Then $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$.

We want shortest paths, so try both and keep whichever is better:

$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right).$$

6 Base case

If $k = 0$, we are not allowed to use any intermediate nodes.

So we must go directly from i to j:

$$d_{ij}^{(0)} = w_{ij}$$

7 Floyd-Warshall algorithm

```
FLOYDWARSHALL(W)
  D = W
  for k = 1, ..., n do
    for i = 1, ..., n do
      for j = 1, ..., n do
        D[i, j] = min (D[i, j], D[i, k] + D[k, j])
      end for
    end for
  end for
  return D
```

8 Not covering here

- Connection between shortest paths and matrix multiplication. (CLRS 25.1)
- Johnson's algorithm, which is faster for sparse graphs. (CLRS 25.3)