Dynamic Programming: All-pairs shortest paths

(Chapter 8)

All-pairs shortest paths

- Problem:
  - Given a directed weighted graph G with n vertices, find the shortest path from any vertex v_i to any other vertex v_j, for all $1 \leq (i,j) \leq n$
- Note: this problem is always solved with an adjacency matrix graph representation
- Applications: This problem occurs in lots of applications – notably in computer games, where it is useful to find shortest paths before planning movement.

Floyd’s algorithm

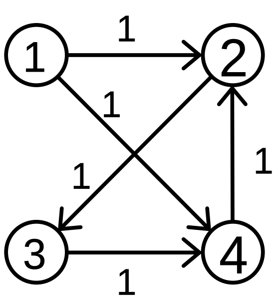
- Consider Warshall’s algorithm, with some changes:
  - Add weight (or cost) to each edge in the initial graph
  - When no edge exists the weight is ∞
    - “You can’t get there from here” (yet)
  - Set the weights on the diagonal to be 0
    - The shortest path from a vertex to itself should be 0

Floyd’s algorithm

- And the real key change:
  - Warshall’s algorithm says this:
    - if (i,k) == (k,j) == 1 then set (i,j) ← 1
    - i.e. If you can get from i to k and from k to j, then now you can get from i to j
  - …but for Floyd’s we will say this:
    - if (i,k) + (k,j) < (i,j) then set (i,j) ← (i,k) + (k,j)
    - i.e. If i-k-j costs less than the (so far) best known path from i to j, then update the best known path"

Floyd's algorithm

- Initial representation of the graph



j

i

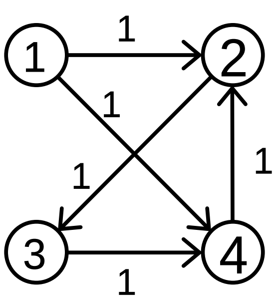
	1	2	3	4
1	0	1	∞	1
2	∞	0	1	∞
3	∞	∞	0	1
4	∞	1	∞	0

Floyd’s Algorithm

Step 1:

- select row 1 and column 1
- for all i,j

  if (i,1) + (1,j) < (i,j) then set (i,j) ← (i,1) + (1,j)



j

i

	1	2	3	4
1	0	1	∞	1
2	∞	0	1	∞
3	∞	∞	0	1
4	∞	1	∞	0

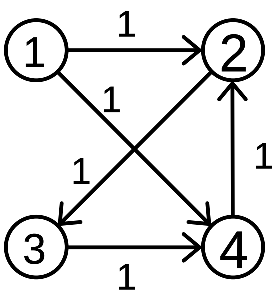
In this case there are no changes.

Floyd’s Algorithm

Step 2:

- select row 2 and column 2
- for all i,j

  if (i,2) + (2,j) < (i,j) then set (i,j) ← (i,2) + (2,j)



j

	1	2	3	4
1	0	1	**2**	1
i 2	∞	0	1	∞
3	∞	∞	0	1
4	∞	1	**2**	0

Notice:

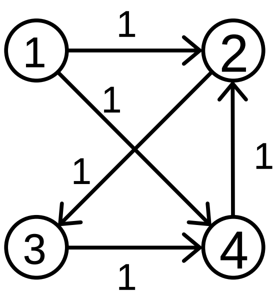
(1,2) + (2,3) < ∞ → set (1,3) ← 2
(4,2) + (2,3) < ∞ → set (4,3) ← 2

Floyd’s Algorithm

Step 3:

- select row 3 and column 3
- for all i,j

  if (i,3) + (3,j) < (i,j) then set (i,j) ← (i,3) + (3,j)



j

i

	1	2	3	4
1	0	1	2	1
2	∞	0	1	**2**
3	∞	∞	0	1
4	∞	1	2	0

There is only one change this time …

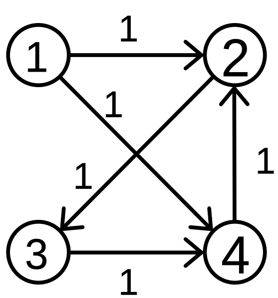
(2,3) + (3,4) < ∞ → set (2,4) ← 2

Floyd’s Algorithm

Step 4:

- select row 4 and column 4
- for all i,j

  if (i,4) + (4,j) < (i,j) then set (i,j) $\leftarrow$ (i,4) + (4,j)



j

i	1	2	3	4
1	0	1	2	1
2	∞	0	1	2
3	∞	**2**	0	1
4	∞	1	2	0

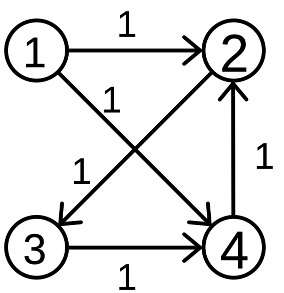
Again, only one change …

$(3,4) + (4,2) < \infty \rightarrow$ set $(3,2) \leftarrow 2$

Floyd’s Algorithm

This time our solution gives the shortest paths from any i to any j.

We can see that the none of 2,3, or 4 have paths to 1, and the algorithm has discovered two hop paths for 1→3, 2→4, 3→2, and 4→3,



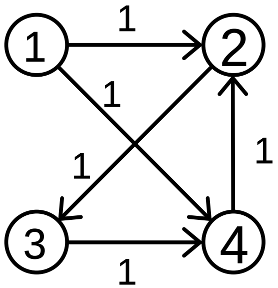
j

i

	1	2	3	4
1	0	1	2	1
2	∞	0	1	2
3	∞	2	0	1
4	∞	1	2	0

Floyd's Algorithm

- The final matrix gives the shortest paths from any i to any j.
- Observations:
  - You can't get from anywhere to 1
  - The algorithm discovered two-hop paths for $1 \to 3$, $2 \to 4$, $3 \to 2$, and $4 \to 3$



j

i	1	2	3	4
1	0	1	2	1
2	∞	0	1	2
3	∞	2	0	1
4	∞	1	2	0

Floyd’s Algorithm (pseudocode)

```
Floyd(G[1..n, 1..n])
    for k ← 1 to n {
        for i ← 1 to n {
            for j ← 1 to n {
                cost_thru_k ← G[i,k] + G[k,j]
                if ( cost_thru_k < G[i,j] ) {
                    set G[i,j] ← thru_k
                }
            }
        }
    }
```

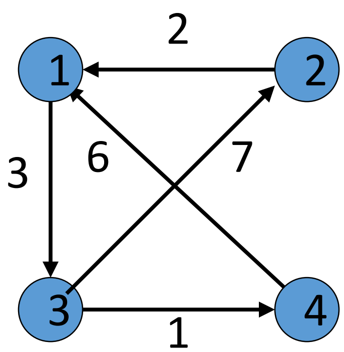
This middle section is referred to as the "Warshall Parameter". We can change it around to solve a variety of problems.

Efficiency: ?

How is this DP?

- (Like Warshall's) the "sub-problem" is that it is finding shortest paths that use vertices 1..k as hopping points
- One new vertex (k) is added into the picture at each step
- After each step, you have a matrix D_k that gives the best (yet) distance through those vertices

Another Example



$$D^{(0)} = \begin{matrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{matrix}$$

$$D^{(1)} = \begin{matrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \mathbf{5} & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \mathbf{9} & 0 \end{matrix}$$

$$D^{(2)} = \begin{matrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \mathbf{9} & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{matrix}$$

$$D^{(3)} = \begin{matrix} 0 & \mathbf{10} & 3 & \mathbf{4} \\ 2 & 0 & 5 & \mathbf{6} \\ 9 & 7 & 0 & 1 \\ 6 & \mathbf{16} & 9 & 0 \end{matrix}$$

$$D^{(4)} = \begin{matrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ \mathbf{7} & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{matrix}$$