Floyd-Warshall APSP algorithm

All Pairs Shortest Paths (APSP)

Given a directed graph G with weight v, the goal is to compute shortest paths (or just their cost) between two vertices $u, v \in V$.

First, if wieghts are non-negative, simply use Dijketra on each possible source vertex $u \in V$. Runtime: $O(V^2 logV + V \cdot E)$

From now on, assume weights can be negative.

We do assume, however, that there are no negative weight cycles.

As a result, shortest path must be simple (i.e. they don't contain cycles or visit some vertexes twice).

In particular, any shortest path must have at most $|V| - 1$ edges.

Three posible algorithms

1. Run Bellman-Ford from each possible source vertex $u \in V$

   runtime: $O(V^2 E)$
2. Floyd-Warshall (today)

   runtime: $O(V^3)$
3. Johnson's algorithm (textbook)

   using reduction to non-negative case, achieving runtime $O(V^2 logV + V \cdot E)$

We assume (without loss of generality) that input is given in matrix form.

$$w(u,v) = \begin{cases} 0 & u = v \\ \text{weight of edge } (u,v) & u \neq v, (u,v) \in E \\ \infty & u \neq v, (u,v) \notin E \end{cases}$$

We will use dynamic programming.

Attemp #1

Subproblems: for each $u, v \in V$, and $k \geq 1$,

$DP(u,v,k) = \text{the weight of shortest path from } u \text{ to } v \text{ using } \neq k \text{ edges.}$

Guess: next-to-last vertex, call it y.

$$DP(u,v,k) = \begin{cases} w(u,v) & k = 1 \\ \min\{DP(u,v,k-1), DP(u,y,k-1) + w(y,v)\} & k > 1 \end{cases}$$

Final output: $DP(u, v, |V| - 1)$

Runtime: #subproblems $\times$ time per subproblem $|V|^3 \times |V| = O(V^4)$

Attemp #2

Idea: only store $DP(u, v, k)$ for k that is a power of 2.

Guess: midpoint y in the optimal path from u to v

$$DP(u,v,k) = \begin{cases} w(u,v) & k=1 \\ \min DP(u,y,\frac{k}{2}), DP(y,v,\frac{k}{2}) & k>1 \end{cases}$$

Runtime: #subproblems $\times$ time per subproblem $|V|^2 log|V| \times |V| = O(|V|^3 log|V|)$

Attemp #3 — Floyd-Warshall

Idea: label vertices $v_1, ..., v_{|V|}$

Subproblems: $\forall u, v \in V, k \in \{0, 1, ..., |V|\}$

$DP(u,v,k) =$
$\text{the weight of shortest path from } u \text{ to } v \text{ only using vertices from } v_1, ..., v_k \text{ as intermediate vertices.}$

$$DP(u,v,k) = \begin{cases} w(u,v) & k=0 \\ \min\{DP(u,v,k-1), DP(u,v_k,k-1) + DP(u_k,v,k-1)\} & k>0 \end{cases}$$

Final output: $DP(u,v,|V|)$ (no restriction)

Runtime: #subproblems $\times$ time per subproblem $|V|^3 \times O(1) = O(V^3)$