COL 351: Analysis and Design of Algorithms

Lecture 16

All-Pairs Distance Computation

Graph type (Directed)	Single-source	All-pairs
Positive edge weights	$O(m + n \log n)$	$O\big((m + n \log n) \cdot n\big)$
Positive/negative edge weights with NO negative-weight-cycle	$O(mn)$	$O\big((mn) \cdot n\big)$

Can we improve this?

Subproblem

Shortest-path(i, j, **S**):
shortest possible path from i to j using internal vertices from set **S**.

Eg.

- $S = \phi$ $\qquad$ $dist(i, j, S) = \begin{cases} wt(i,j) & \text{if } (i,j) \in E \\ \infty & \text{o/w} \end{cases}$

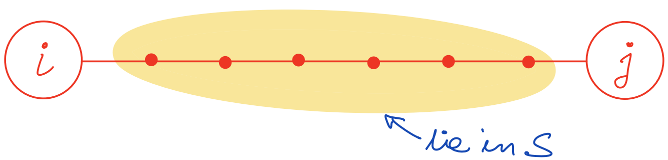
- $S = V$ $\qquad$ $dist(i, j, S) =$ Distance from i to j in G.

Subproblem

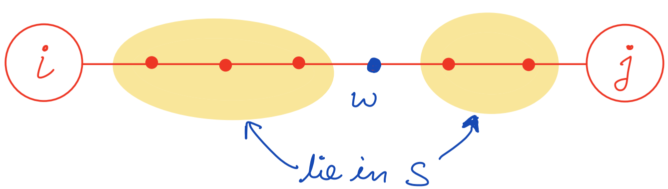
Shortest-path(i, j, **S**):
shortest possible path from i to j using internal vertices from set **S**.

> ***Question:***
> Given Shortest-path(i, j, **S**) for all-pairs, find Shortest-path(i, j, **S** U {w}).

CASE 1 shortest possible path doesn't contain "w"



CASE 2



$$\text{DIST}(i,j,S \cup w) = \min \begin{cases} \text{DIST}(i,j,S), & \text{(case 1)} \\ \text{DIST}(i,w,S) + \text{DIST}(w,j,S) & \text{(case 2)} \end{cases}$$

Floyd-Warshall Algorithm

Algorithm:

1. Create 2-D array distance of size $n \times n$ with all entries initialised to ∞.

2. for each edge (x, y) do
   distance$[x, y] \leftarrow$ weight(x, y)

3. for each vertex v do
   distance$[v, v] \leftarrow 0$

4. for k =1 to n: S here is $\{1, .., k-1\}$
   for i, j =1 to n :
   distance$[i, j]$ = min{distance$[i, j]$, distance$[i, k]$ + distance$[k, j]$ }

Time = $O(n^3)$

Space = $O(n^2)$

Remark:
The order in which pairs (i, j) (i, k) & (k, j) are processed is NOT important.

Correctness

4. for k =1 to n:
   for i, j =1 to n :
   distance[i, j] = min{distance[i, j] , distance[i, k] + distance[k, j] }

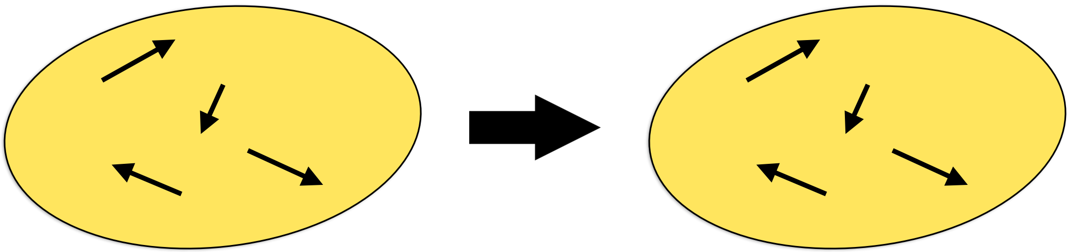
Invariant : Before beginning iteration k :

distance[i, j] stores shortest possible path length from i to j when internal vertices are restricted from set $[1, k\text{-}1]$.

Johnson's Approach

$$O(n^3) \longrightarrow O(mn \log n)$$

Transform $G = (V, E, wt)$ to a new graph $H = (V, E, wt^*)$



$G = (V, E, wt)$ $H = (V, E, wt^*)$

Edge weights of H ***are non-negative***

Such that, $\forall (x, y)$:

$$(x, y)\text{-shortest-path-in-}G \equiv (x, y)\text{-shortest-path-in-}H$$

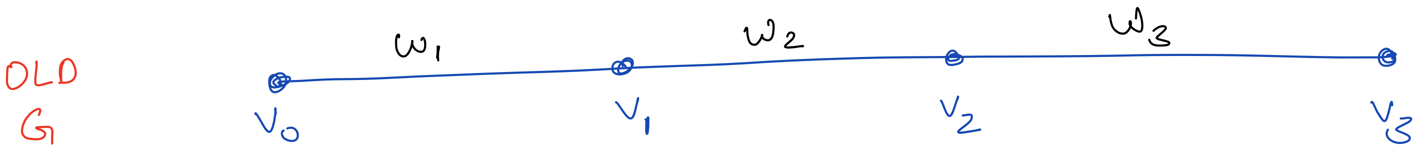
What should be the new weight function if we want shortest-paths remain intact?

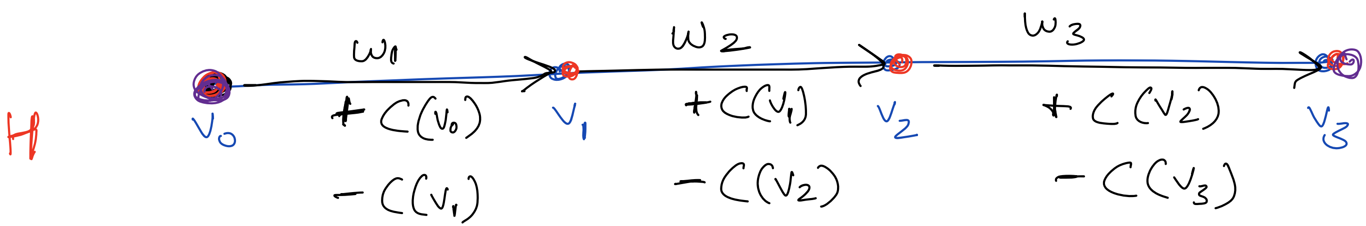
$$\boxed{wt^*(x, y) = C(x) + wt(x, y) - C(y)}$$

$C(x)$

$x \longrightarrow y$

$-C(y)$





$$\text{new-wt}(P) = \text{old-wt}(P) + C(v_0) - C(v_3)$$

New weight function

$\boxed{C(v) = dist_G(s, v)}$

$s \leftarrow$ *an arbitrary vertex in* V

$$\boxed{wt^*(x, y) = dist_G(s, x) + wt(x, y) - dist_G(s, y)}$$

Ques: Is $wt^*(x, y) \geq 0$, for all edges (x, y)?

Ans: for all edges (x, y)

$$dist_G(s, y) \leq dist_G(s, x) + wt(x, y)$$

Johnson's Algorithm

$s \leftarrow$ an arbitrary vertex in V

For Each $v \in V$:
    compute $dist_G(s, v)$ } $O(mn)$ # Bellman Ford

For Each $(x, y) \in E$:
    $wt^*(x, y) = dist_G(s, x) + wt(x, y) - dist_G(s, y)$

Compute $H = (V, E, wt^*)$

For Each $(a, b) \in V \times V$:
    compute $dist_H(a, b)$ } $O(mn + n^2 \log n)$ # n runs of Dijkstra's algo

For Each $(a, b) \in V \times V$:
    $dist_G(a, b) = \cancel{dist_G(s, a)} + dist_H(a, b) \cancel{- dist_G(s, b)}$ } $O(n^2)$

$- dist_G(s, a)$ $+ dist_G(s, b)$

Return $dist_G$.

Total time $= O(mn + n^2 \log n)$