Shortest paths

Problem: Given a (directed) weighted graph G, compute the min length path from s to t (shortest path), where the length of a path is the sum of its edge weights.

$m := \#$ edges, $n := \#$ vertices

COS 226:

Dijkstra: start with $d[s]=0$, $d[x]=\infty$ for $x \neq s$.
In each round, find x with minimum $d[x]$ that is unprocessed, process x by updating dist using x: $d[y] \leftarrow \min(d[y], d[x]+w(x,y))$.
With heap: $O(m \log n)$. Works if $w(x,y) \geq 0$.

Bellman-Ford: In each round, update the distances using all edges. Run for n rounds
$O(m \cdot n)$

Today: All pairs shortest path (APSP).

Compute $\forall x,y$, the length of the shortest path from x to y. Allow negative weights.

Lemma: There exists a SP with no repeated vertices. (from 226)

Dynamic Programming

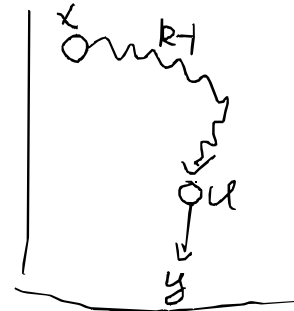
DP based on # edges on the SP

Let $d_{xy}^{(k)} :=$ the length of the shortest path from x to y using at most k edges

Base case: $d_{xy}^{(1)} = \begin{cases} w(x,y) & \text{if } (x,y) \text{ edge} \\ \infty & \text{o.w.} \end{cases}$ assuming $w(x,x) = 0$

To compute $d_{xy}^{(k)}$, guess the last edge in the shortest path.

$$d_{xy}^{(k)} = \min_u \{d_{xu}^{(k-1)} + w(u,y)\}$$



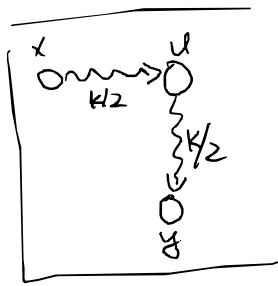
Output $\{d_{xy}^{(n)}\}$.

Bellman-Ford

Time complexity: $O(n^2 m)$ (same as n times)

A better solution: guess the midpoint.

$$d_{xy}^{(k)} = \min_u \left\{ d_{xu}^{(k/2)} + d_{uy}^{(k/2)} \right\}.$$



(for even k)

Compute $d_{xy}^{(1)}$ (forall x, y)

then $d_{xy}^{(2)}, d_{xy}^{(4)}, \ldots d_{xy}^{(2^i)}, \ldots$

until $2^i > n$.

Time: $O(n^3 \log n)$.

cannot use this trick to improve Bellman-Ford.

need $d_{uy}^{(k/2)}$ to compute $d_{xy}^{(k)}$, a different source.

An equivalent view using Min-Plus matrix multi

Let A, B be $n \times n$ matrices.

Define Min-Plus product of A and B as

$$(A \odot B)_{ij} = \min_k \{A_{ik} + B_{kj}\}.$$

Easy to verify: $A \odot (B \odot C) = (A \odot B) \odot C$

Let $D^{(k)}$ be matrix s.t. $D^{(k)}_{xy} = d^{(k)}_{xy}$.

W be ... $W_{xy} = d^{(1)}_{xy}$.

Then $D^{(k)} = D^{(k-1)} \odot W$.

and $D^{(n)} = W^{\odot n}$

Compute $W, W^{\odot 2}, W^{\odot 4}, \ldots, W^{\odot 2^i}, \ldots$.

Matrix Multi takes $O(n^3)$ time.

Time: $O(n^3 \log n)$.

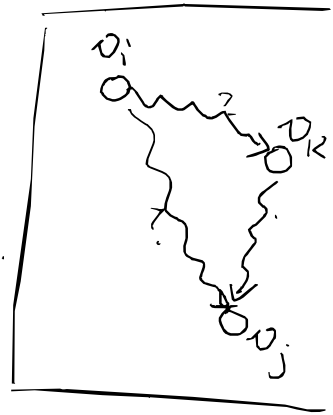
Floyd-Warshall

Fix a vertex order $v_1, \ldots, v_n$.

Let $C^{(k)}_{ij}$ be the length of the shortest path from v_i to v_j only going through intermediate vertices in $\{v_1, \ldots, v_k\}$.

Base case: $C_{ij}^{(0)} = w(v_i, v_j)$

To compute $C_{ij}^{(k)}$, guess if the shortest goes through v_k



$$C_{ij}^{(k)} = \min\left\{ C_{ij}^{(k-1)}, \underbrace{C_{ik}^{(k-1)} + C_{kj}^{(k-1)}}_{\text{shortest path is simple.}} \right\}$$

Output $C_{ij}^{(n)}$.

Time: $O(n^3)$.

Johnson's algorithm

Let $f: V \to \mathbb{R}$ be a weight function.

Let G' be the graph with weights

$$w'(u,v) = w(u,v) + f(u) - f(v).$$

Lemma: Shortest paths in G are the same as the shortest paths in G'.

Proof. PSet 1. P3(b).

Claim: A function f s.t. $w'(u,v) \geq 0 \ \forall u, v$.
can be computed in $O(mn)$ time.

Proof: Bellman-Ford + P3(c).

Then G' has nonnegative weights.

Run Dijkstra from every vertex.

Time: $O(mn \log n)$
or $O(mn + n^2 \log n)$ [using Fibonacci heap).