All-Pairs Shortest Paths

Given: Digraph $G=(V,E)$, where $V=\{1,2,\ldots,n\}$, possibly negative costs $c(i,j)$, BUT no negative cycles! ($c(i,j) = \infty$ means no edge $(i,j)$ in $G$)

Compute: $D(i,j) =$ cost of cheapest path from $i$ to $j$, for all $i,j$ in $V$.

Later, will also want an algorithm that, given $(i,j)$, finds a cheapest path from $i$ to $j$.

Observation: Every cheapest path from $i$ to $j$ must be simple, i.e., with no cycles!

Floyd-Warshall DP algorithm

Step 1: Array

$$A(k,i,j)$$

Step 2: Recurrence

$$A(0,i,i) = 0, \forall i$$
$$A(0, i, j) = c(i,j)$$

$$A(k, i, j) = \min \{ A(k-1, i, j), A(k-1, i, k') + A(k-1, k', j) \}$$

Case 1: node $k$ is not used

node $k$ is not used

node $k$ is not used

node $k$ is not used

node $k$ is not used
Step 3: Algorithm to fill in the array.

array $A[K, i, j]$, $0 \leq K \leq n$,
$1 \leq i \leq n$,
$1 \leq j \leq n$

Runtime: $O(n^3)$

Given $(i, j)$ points out cheapest

$x(i, j) = A[n, i, j]$

$\text{time: } O(1(n))$

Step 4: Recover shortest paths from the array

PrintOpt($K$, $i$, $j$) % Print Opt

if $K = 0$ % base case
then if $i = j$ then return
else return edge $(i, j)$

root call

If $K = 0$ % base case
then if $i = j$ then return
else return edge $(i, j)$
Shortest paths

Shortest path problem. Given a digraph $G = (V,E)$, with arbitrary edge weights or costs $c_{uv}$, find cheapest path from node $s$ to node $t$.

cost of path = $9 - 3 + 1 + 11 = 18$
Shortest paths: failed attempts

**Dijkstra.** Can fail if negative edge weights.

**Reweighting.** Adding a constant to every edge weight can fail.

Negative cycles

**Def.** A negative cycle is a directed cycle such that the sum of its edge weights is negative.
Shortest paths and negative cycles

**Lemma 1.** If some path from \( v \) to \( t \) contains a negative cycle, then there does not exist a cheapest path from \( v \) to \( t \).

**Pf.** If there exists such a cycle \( W \), then can build a \( v \rightarrow t \) path of arbitrarily negative weight by detouring around cycle as many times as desired. ■

\[ c(W) < 0 \]

---

Shortest paths and negative cycles

**Lemma 2.** If \( G \) has no negative cycles, then there exists a cheapest path from \( v \) to \( t \) that is simple (and has \( \leq n-1 \) edges).

**Pf.**
- Consider a cheapest \( v \rightarrow t \) path \( P \) that uses the fewest number of edges.
- If \( P \) contains a cycle \( W \), can remove portion of \( P \) corresponding to \( W \) without increasing the cost. ■

\[ c(W) \geq 0 \]
Shortest path and negative cycle problems

**Shortest path problem.** Given a digraph $G=(V,E)$ with edge weights $c_{vw}$ and no negative cycles, find cheapest $v\to t$ path for each node $v$.

**Negative cycle problem.** Given a digraph $G=(V,E)$ with edge weights $c_{vw}$, find a negative cycle (if one exists).

![Shortest-paths tree and negative cycle]

---

**Shortest paths: dynamic programming**

**Def.** $OPT(i, v) = \text{cost of shortest } v\to t \text{ path that uses } \leq i \text{ edges.}$

- **Case 1:** Cheapest $v\to t$ path uses $\leq i - 1$ edges.
  - $OPT(i, v) = OPT(i-1, v)$

- **Case 2:** Cheapest $v\to t$ path uses exactly $i$ edges.
  - if $(v, w)$ is first edge, then $OPT$ uses $(v, w)$, and then selects best $w\to t$ path using $\leq i - 1$ edges

$$OPT(i, v) = \begin{cases} \infty & \text{if } i = 0 \\ \min \{ OPT(i-1, v), \min_{(v, w) \in E} \{ OPT(i-1, w) + c_{vw} \} \} & \text{otherwise} \end{cases}$$

**Observation.** If no negative cycles, $OPT(n-1, v) = \text{cost of cheapest } v\to t \text{ path}.$

**Pf.** By Lemma 2, cheapest $v\to t$ path is simple. •
Shortest paths: implementation

**Algorithm: Shortest Paths** $(V, E, c, t)$

**FOR** each node $v \in V$

- $M[0, v] \leftarrow \infty.$
- $M[0, t] \leftarrow 0.$

**FOR** $i = 1$ **TO** $n - 1$

**FOR** each node $v \in V$

- $M[i, v] \leftarrow M[i-1, v].$

**FOR** each edge $(v, w) \in E$

- $M[i, v] \leftarrow \min \{ M[i, v], M[i-1, w] + c_{vw} \}.$

**Theorem 1.** Given a digraph $G = (V, E)$ with no negative cycles, the dynamic programming algorithm computes the cost of the cheapest $v \rightarrow t$ path for each node $v$ in $\Theta(mn)$ time and $\Theta(n^2)$ space.

**Pf.**
- Table requires $\Theta(n^2)$ space.
- Each iteration $i$ takes $\Theta(m)$ time since we examine each edge once.

**Finding the shortest paths.**
- **Approach 1:** Maintain a $\text{successor}(i, v)$ that points to next node on cheapest $v \rightarrow t$ path using at most $i$ edges.
- **Approach 2:** Compute optimal costs $M[i, v]$ and consider only edges with $M[i, v] = M[i-1, w] + c_{vw}.$

$$\text{OPT}(i, v) = \text{OPT}(i-1, v) \lor v \quad (\ast)$$

$$\text{OPT}(i, \cdot) = \text{E}(\text{OPT}(i-1, \cdot))$$
Detecting negative cycles

**Negative cycle detection problem.** Given a digraph \( G = (V, E) \), with edge weights \( c_{vw} \), find a negative cycle (if one exists).
Detecting negative cycles: application

**Currency conversion.** Given \( n \) currencies and exchange rates between pairs of currencies, is there an arbitrage opportunity?

**Remark.** Fastest algorithm very valuable!

---

**Detecting negative cycles**

**Lemma 5.** If \( OPT(n, v) = OPT(n - 1, v) \) for all \( v \), then no negative cycle can reach \( r \).

**Pf.** Bellman-Ford algorithm.

**Lemma 6.** If \( OPT(n, v) < OPT(n - 1, v) \) for some node \( v \), then (any) cheapest path from \( v \) to \( r \) contains a cycle \( W \). Moreover \( W \) is a negative cycle.

**Pf.** [by contradiction]

- Since \( OPT(n, v) < OPT(n - 1, v) \), we know that shortest \( v \rightarrow r \) path \( P \) has exactly \( n \) edges.
- By pigeonhole principle, \( P \) must contain a directed cycle \( W \).
- Deleting \( W \) yields a \( v \rightarrow r \) path with < \( n \) edges \( \Rightarrow \) \( W \) has negative cost.

---

**Detecting negative cycles**

**Theorem 4.** Can find a negative cycle in \( \Theta(nm) \) time and \( \Theta(n^2) \) space.

**Pf.**

- Add new node \( r \) and connect all nodes to \( r \) with 0-cost edge.
- \( G \) has a negative cycle iff \( G' \) has a negative cycle than can reach \( r \).
- If \( OPT(n, v) = OPT(n - 1, v) \) for all nodes \( v \), then no negative cycles.
Add new node \( n \) and connect all nodes to \( n \) with \( 0 \)-cost edges.

- \( G \) has a negative cycle iff \( G' \) has a negative cycle that can reach \( n \).
- If \( OPT(n, v) = OPT(n - 1, v) \) for all nodes \( v \), then no negative cycles.
- If not, then extract directed cycle from path from \( v \) to \( n \).

(cycle cannot contain \( n \) since no edges leave \( n \))