COMPSCI 311: Introduction to Algorithms
Lecture 15: Dynamic Programming – Shortest Paths
Marius Minea
University of Massachusetts Amherst
slides credit: Dan Sheldon (adapted)

Currency Trading

▶ Problem:
given directed graph with exchange rate $r_e$ on edge $e$, find best exchange rate $s \rightarrow t$, i.e., path $P$ with maximum product $\prod_{e \in P} r_e$ over edges

▶ Assumption (no arbitrage):
no cycles $C$ with $\prod_{e \in C} r_e > 1$.

Shortest Paths
▶ We know how to find minimum sum, not maximum product, but
▶ logarithm of product is sum of logs
▶ maximize $x$ means minimize $-x$
▶ Let $c_e = -\log r_e$ be the cost of edge $e$
▶ Let the path cost be the negative log of the path exchange rate.

$\text{cost}(P) = -\log \prod_{e \in P} r_e$
$= \sum_{e \in P} (-\log r_e)$
$= \sum_{e \in P} c_e$

▶ Equivalent problem: find the $s \rightarrow t$ path of minimum cost

Currency Trading

▶ Negative edge weights!
▶ Problem: given a graph with possibly negative edge weights, find shortest $s \rightarrow t$ path
▶ Assumption: no cycle $C$ with $\sum_{e \in C} c_e < 0$. Why?

Dijkstra’s Algorithm: Negative Edge Behavior

What is the shortest path value the algorithm finds for $d(s, v)$?

Clicker Question 1

When run on a graph with negative edges, Dijkstra’s algorithm:

A: Does not give the right value if shortest path has negative edge.
B: May give the right value even if the shortest path has a negative edge.
C: Does not give the right value if the target node is first reached through a positive edge.
D: Gives the right value if the target node is first reached through a negative edge.

Choose the most precise answer!
Bellman-Ford Algorithm: Setup

Consider shortest paths from any node to a given target node \( t \) (single-destination shortest paths).

Like single-source, but destination more relevant e.g., in routing.

Consider paths with increasing number of edges to target.

**Fact.** If no negative cycles, shortest path has at most \( n - 1 \) edges.

Why?

Path with \( \geq n \) edges has \( \geq n + 1 \) nodes: would repeat some node, thus cycle!

---

Clicker Question 2

In the following graph, which is the value of the shortest \( s \to t \) path found by Dijkstra’s algorithm?

![Graph](image)

A: 26  
B: 20  
C: 12  
D: 11

---

Clicker Question 3

For shortest paths from any \( v \) to a fixed \( t \), we’d like to compute \( \text{OPT}(i + 1, v) \) from \( \text{OPT}(i, v) \), by incrementing the edge count \( i \).

If we find a better path starting with edge \((v, w)\), we want to update

\[
\text{OPT}(i + 1, v) = c_{v, w} + \text{OPT}(i, w)
\]

Should \( \text{OPT}(i, v) \) mean the optimal cost from \( v \) to \( t \)?

A: on a path with \( i \) edges  
B: on a path with at most \( i \) edges

---

Bellman-Ford Algorithm

\[
\text{OPT}(i, v) = \min \left\{ \text{OPT}(i - 1, v), \min_{w \in V} \{c_{v, w} + \text{OPT}(i - 1, w)\} \right\}
\]

**Shortest-Path** \((G, s, t)\)

\( n = \) number of nodes in \( G \)

Create array \( M \) of size \( n \times n \)

Set \( M[0, t] = 0 \) and \( M[0, v] = \infty \) for all other \( v \)

for \( i = 1 \) to \( n - 1 \) do

for all nodes \( v \) in any order do

Compute \( M[i, v] \) using the recurrence above

end for

end for

Running time? \( O(n^3) \). Better analysis: \( O(mn) \).

---

Shortest paths with negative weights: practical improvements

**Space optimization.** Maintain two 1D arrays (instead of 2D array).

* \( d(v) \) = length of a shortest \( v \to t \) path that we have found so far.

* \( \text{successor}(v) = \) next node on a \( v \to t \) path.

**Performance optimization.** If \( d(w) \) was not updated in iteration \( i - 1 \), then no reason to consider edges entering \( w \) in iteration \( i \).
Clicker Question 4

Consider a directed graph with arbitrary edge weights.

Then, at every step of running Bellman-Ford

A: Following successor[v] pointers gives a \( v \rightarrow t \) path
B: The length of the successor[v] path is \( d[v] \)
C: Both A and B
D: Neither A nor B

A: No, \( d[v] \) can be one iteration behind, if successor[v] = w same, but \( d[w] \) just got updated.
B: No, for negative-weight cycles (next slide)

Detecting Negative-Weight Cycles

We’ve seen that absent negative-weight cycles, a shortest path has at most \( n-1 \) edges.

Run for one extra iteration \( (n) \). If \( \text{OPT}(n,v) \) decreases for some \( v \), we have a negative-weight cycle! (why?)

But this is only over paths to a fixed target node \( t \). How to cover the entire graph?

Add dummy sink node with zero-cost edges from all nodes.
Use this as target (all nodes are predecessors, will be covered).

Detecting negative cycles

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

**PF.**
- Add new sink node \( t \) and connect all nodes to \( t \) with 0-length edge.
- \( G' \) has a negative cycle iff \( G' \) has a negative cycle.
- Case 1. [ \( \text{OPT}(n,v) = \text{OPT}(n-1,v) \) for every node \( v \) ]
  - By Lemma 7, no negative cycles.
- Case 2. [ \( \text{OPT}(n,v) < \text{OPT}(n-1,v) \) for some node \( v \) ]
  - Using proof of Lemma 8, can extract negative cycle from \( v \rightarrow \) path.

Detecting negative cycles

**Theorem 5.** Can find a negative cycle in \( O(mn) \) time and \( \Theta(n) \) extra space.

**PF.**
- Run Bellman–Ford–Moore on \( G' \) for \( n' = n+1 \) passes (instead of \( n'-1 \)).
- If no \( d[v] \) values updated in pass \( n' \), then no negative cycles.
- Otherwise, suppose \( d[t] \) updated in pass \( n' \).
- Define \( \text{pass}(v) = \text{last pass in which } d[v] \text{ was updated} \).
- Observe \( \text{pass}(v) = n' \) and \( \text{pass}(\text{successor}[v]) \geq \text{pass}(v) - 1 \) for each \( v \).
- Following successor pointers, we must eventually repeat a node.
- Lemma 6 \( \Rightarrow \) the corresponding cycle is a negative cycle.

**Remark.** See p. 304 for improved version and early termination rule. (Tarjan’s subtree disassembly trick)