DS 320 Algorithms for Data Science **Lecture #16**
Spring 2026 Prof. Kira Goldner and Jeffrey Considine

Dynamic Programming IV: Shortest Paths

The Problem

Given a graph $G = (V, E)$ with *any edge weights* w_e for all $e \in E$, a source s, output the shortest $s - v$ path for every $v \in V$.

Naïve Approach

Let $N(v)$ be the neighbors of v.

- Subproblem: Let $\text{OPT}(v)$ denote the length of the shortest path from s to v.
- Recurrence: $\text{OPT}(v) = \min_{u \in N(v)} \text{OPT}(u) + w_{(u,v)}$.
- Base Case: $\text{OPT}(s) = 0$.

Issues:

- If there's a negative cycle, base case is actually wrong.
- What is the graph of subproblem dependencies?
- Actually just Dijkstra, sliced wrong.

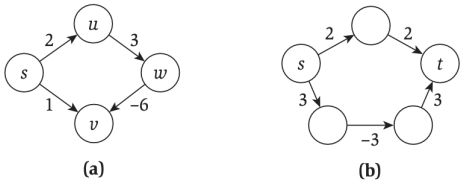


Figure 1: From Kleinberg Tardos. (a) With negative edge costs, Dijkstra's Algorithm can give the wrong answer for the Shortest-Path Problem. (b) Adding 3 to the cost of each edge will make all edges non-negative, but it will change the identity of the shortest $s - t$ path.

The Bellman-Ford Algorithm

Main Idea: Impose a measure of progress—parametrize the subproblems.

More specifically:

- Consider a shortest path from $s \to u \to v \to t$ with k edges

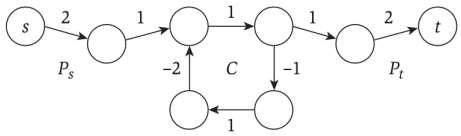


Figure 2: From Kleinberg Tardos. In this graph, one can find $s - t$ paths of arbitrarily negative cost (by going around the cycle C many times).

- $s \to u \to v$ is a shortest $s \to v$ path with $k - 1$ edges

Measure of progress: the number of edges in path. First compute all shortest paths with ≤ 1 edges. Then shortest paths with ≤ 2 edges. And so on, until we compute shortest paths of length $\leq n-1$.

Structural Observation: If a G has no negative cycles, there's a shortest path from s to t with $\leq n - 1$ edges.

Proof sketch. By contradiction. Suppose G has no negative cycles and there is some t such that the shortest path is of length $k > n - 1$.

- Take an $s \to t$ path P with $k > n - 1$ edges.
- Due to its length, P must contain some cycle C.
- By the supposition, C's weight is nonnegative.
- Consider the path $P' = P \setminus C$.
  - P' is an $s \to t$ path.
  - weight$(P') \leq$ weight(P).

Since we can do this for any path of length $k > n - 1$, we have a contradiction to it being the shortest. □

Formal Description

State Your Subproblem: Let $\text{OPT}(v, i)$ denote the length of the shortest $s \to v$ path with at most i edges.

State Your Recurrence: $\text{OPT}(v, i) = \min\left\{\text{OPT}(v, i-1), \min_{u \in N(v)} \left[\text{OPT}(u, i-1) + w_{(u,v)}\right]\right\}$

Prove Your Recurrence: Say you're constructing a shortest $s \to v$ path of length $\leq i$. What are your options? Either best path has length i, OR the length $\leq i - 1$.

- Best path has length $\leq i - 1$: $\text{OPT}(v, i) = \text{OPT}(v, i - 1)$.
- Best path has length i:
  - (Want to write in terms of paths of length $\leq i - 1$)

- Best path is $s \to u \to v$ for some u
- Path length is $\text{OPT}(u, i-1) + w_{(u,v)}$.
- (Choose the best u.)

- (Choose best of two options for path length.)

State Your Base Cases: $\text{OPT}(s, 0) = 0$ and $\text{OPT}(v, 0) = \infty$ for $v \neq s$.

Present Your Algorithm:

Algorithm 1 ALG

Input:
Memo[ ][ ] = new int[n][n]
Memo[s][0] = 0
for all $v \neq s$ **do**
  Memo[v][0] = ∞
end for
for i from 1 to $n-1$ **do**
  for all v **do**
    Memo[v][i] = min $\Big\{$Memo[v][$i-1$], $\min_{u \in N(v)}$ Memo[u][$i-1$] + $w_{(u,v)}$]$\Big\}$
  end for
end for
return Memo[t][$n-1$]

*If there are no negative cycle, this will be correct.

Runtime:

- Size of table: $O(n^2)$.
- Time to fill in table:

  (How many table lookups?) $O(n)$.

Total time: $O(n^3)$. (*Run on example.*)

Better Runtime Analysis

- How many row updates? $O(n)$
- How many table lookups per row update? $O(m)$
- Runtime = $O(mn)$ (When m is small, algorithm is faster.)

Detecting Negative Cycles

Idea: Add an extra column (Column n) to the memo table.

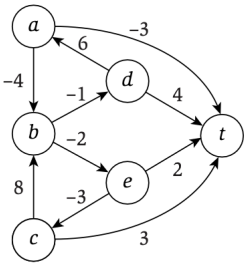
Claim 1. There is a negative cycle in the graph if and only if Column $n \neq$ Column $n-1$.

Proof. ($\Leftarrow$) Suppose the graph has no negative cycles. Then by the Structural Observation, the extra column must be the same as the previous one. That is, when there are no negative cycles, then Column $n =$ Column $n-1$.

($\Rightarrow$) Suppose there is a negative cycle in the graph.. Then it is the case that $\text{OPT}(v, i) \to -\infty$ for some v. Suppose we add column n. If it were the same as column $n-1$, then all following columns would also be the same. However, we know that at least one entry goes to $-\infty$, so it must be that when there is a negative cycle, Column $n \neq$ Column $n-1$. □

To detect negative cycles:

- Run Bellman-Ford, add an nth column.
- If Column $n =$ Column $n-1$, return "No Negative Cycles"
- Otherwise, return that there's a negative cycle.



(a)

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	−3	−3	−4	−6	−6
b	∞	∞	0	−2	−2	−2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

(b)

Figure 3: From Kleinberg Tardos. For the directed graph in (a), the Shortest-Path Algorithm constructs the dynamic programming table in (b).

Space-Efficient Bellman-Ford

How much space does Bellman-Ford use? $O(n^2)$

But notice that every column only depends on the previous column, so we can actually throw away the earlier columns.

Total space usage: $O(n)$.

Algorithm 2 Alg(j)

Input:
Memo[ ] = new int[n]
Memo[s] = 0
for all $v \neq s$ **do**
    Memo[v] = ∞
end for
for i from 1 to $n-1$ **do**
    update(Memo)
end for

Algorithm 3 update(Memo)

Input:
Temp = Memo
Memo[v] = $\min\left\{\text{Temp}[v], \min_{u \in N(v)} \text{Temp}[u] + w_{(u,v)}]\right\}$