

## 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:  $\text{OPT}(s) = 0$ .

What's wrong with this approach? List any issues you see:

## The Bellman-Ford Algorithm

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

More specifically:

- Consider a shortest path from  $s \rightarrow u \rightarrow v \rightarrow t$  with  $k$  edges
- $s \rightarrow u \rightarrow v$  is a shortest  $s \rightarrow v$  path with  $k - 1$  edges

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

### Structural Observation:

*Proof.*

**Formal Description**

**State Your Subproblem:**

**State Your Recurrence:**

**Prove Your Recurrence:**

**State Your Base Cases:**

**Present Your Algorithm:**

**Runtime:**

- Size of table:
- Time to fill in table:

(How many table lookups?)

Total time:

### **Better Runtime Analysis**

- How many row updates?
- How many table lookups per row update?
- Runtime =

### **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,

( $\Rightarrow$ ) Suppose there is a negative cycle in the graph.

To detect negative cycles:

- Run Bellman-Ford, add an  $n$ th column.
- If Column  $n =$  Column  $n - 1$ , return “No Negative Cycles”
- Otherwise, return that there’s a negative cycle.

### **Space-Efficient Bellman-Ford**

How much space does Bellman-Ford use?

**Observation:**

**Improved algorithm:**

Total space usage: