CS 7800: Advanced Algorithms

Class 6: Dynamic Programming III

- Knapsack
- Shortest Paths

Jonathan Ullman
September 23, 2025

Housekeeping

- HW1 is returned
- HW2 due Friday (can use up to 2 late days)

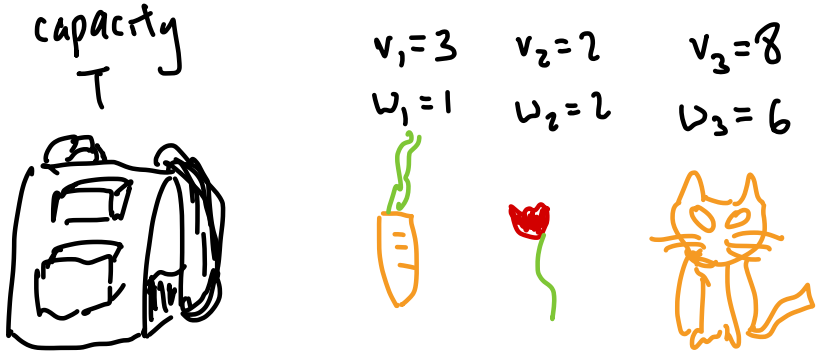
- OH tomorrow → on Zoom 2-3pm
  ↳ will also add slots

- EXAM I TUESDAY OCT 7th

- HW3 out this Friday due next Friday 10/3

The Knapsack Problem

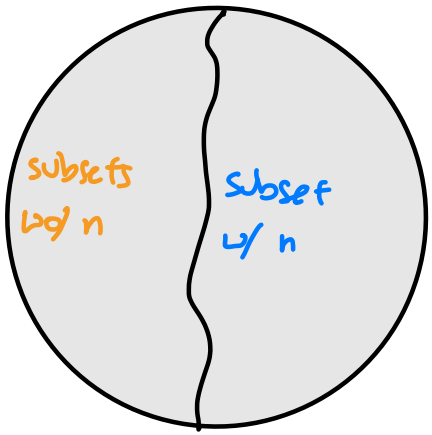
- **Input:** n items for your knapsack with value v_i and weight w_i and a capacity $T \in \mathbb{N}$ a non-negative integer
- **Output:** the most valuable subset of items that fits in the knapsack
  - **subset** $S \subseteq \{1, \dots, n\}$
  - **value** $V_S = \sum_{i \in S} v_i$ as large as possible
  - **weight** $W_S = \sum_{i \in S} w_i$ at most T



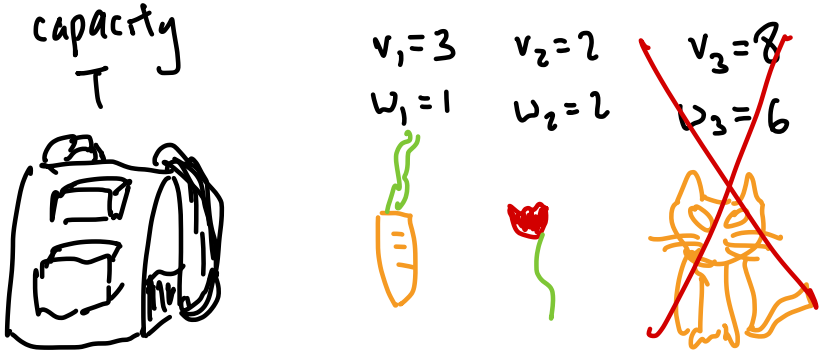
Writing the Recurrence

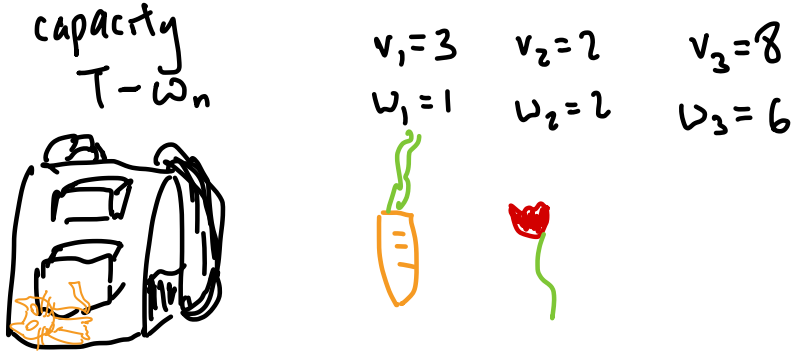
$OPT(i)$ = value of the best subset among items $\{1, \ldots, i\}$ $\quad i = 0, 1, \ldots, n$

subproblems



$$OPT(n) = \max \left\{ \begin{array}{l} \text{value of best solution w/o } n \\ \downarrow \\ OPT(n-1) \end{array} \right., \left. \begin{array}{l} \text{value of best solution w/ } n \\ v_n + OPT(n-1) \ ? \end{array} \right\}$$

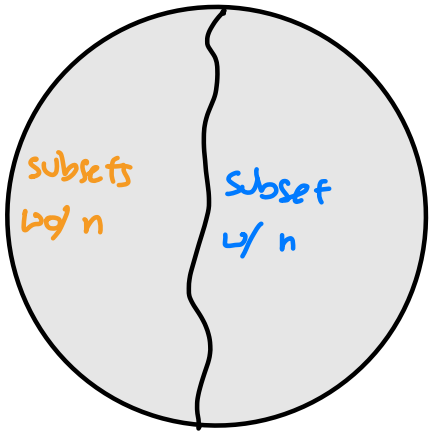




Writing the Recurrence

$OPT(i, S)$ = maximum value of a subset of items $\{1, \ldots, i\}$ with cap. $\leq S$ $\quad i = 0, 1, \ldots, n$ $\quad S = 0, 1, \ldots, T$

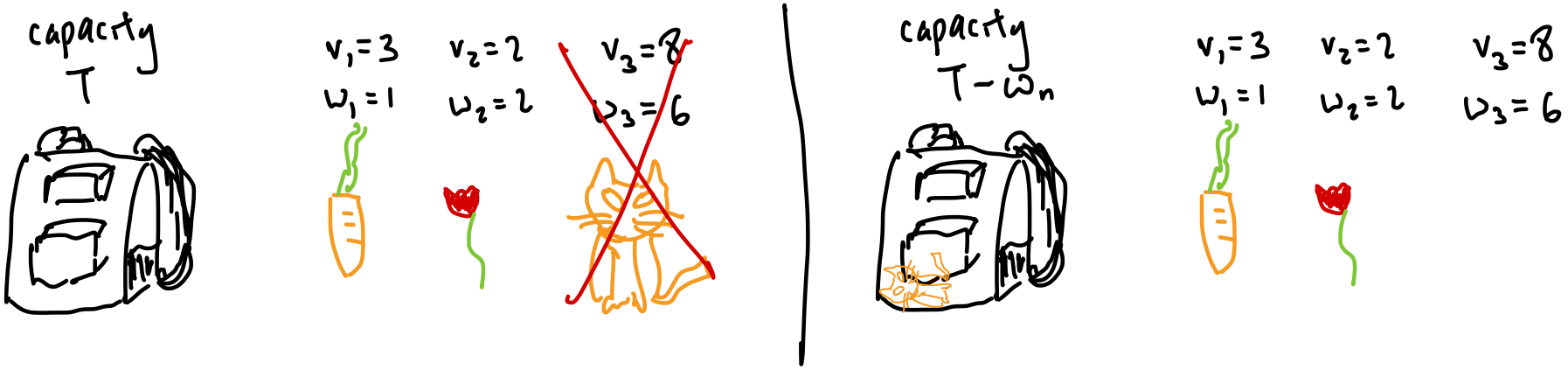
subproblems $\approx nT$ subproblems



$$OPT(n, T) = \max \left\{ \begin{array}{l} \text{value of best solution w/o } n \\ \text{value of best solution w/ } n \end{array} \right\}$$

$$OPT(n, T) = \max \{ OPT(n-1, T),\ v_n + OPT(n-1, T - w_n) \}$$

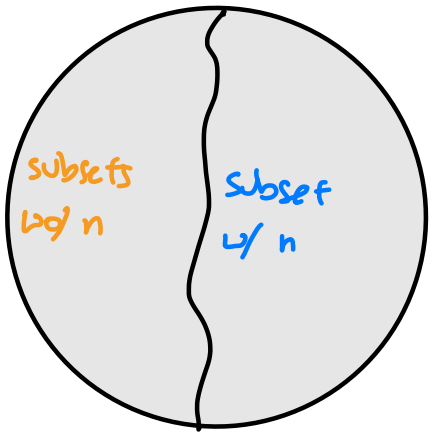
$OPT(0, T) = 0$ $\quad OPT(i, 0) = 0$



Writing the Recurrence

$OPT(i, S)$ = maximum value of a subset of items $\{1, \ldots, i\}$ with cap. $\leq S$ $\quad i = 0, 1, \ldots, n$ $\quad S = 0, 1, \ldots, T$

subproblems $\approx nT$ subproblems



$OPT(n, T) = \max \{$ value of best solution w/o n , value of best solution w/ n $\}$

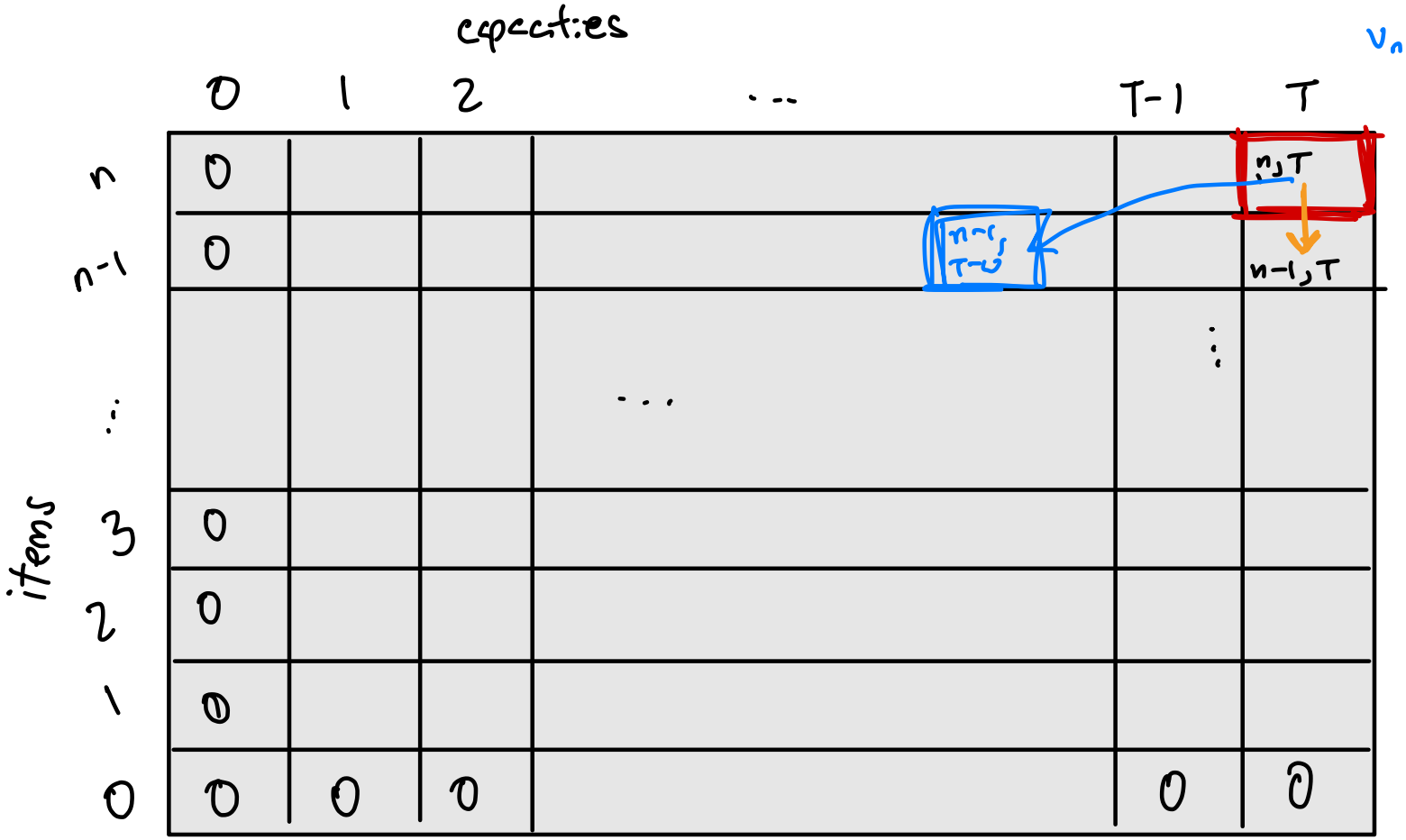
$$OPT(n, T) = \begin{cases} OPT(n-1, T) & \text{if } w_n > T \\ \max \{ OPT(n-1, T),\ v_n + OPT(n-1, T - w_n) \} & \text{if } w_n \leq T \end{cases}$$

$OPT(0, T) = 0$ $\quad OPT(i, 0) = 0$

Solving the Recurrence

$OPT(i, S)$ = optimal value items $\{1, \ldots, i\}$ cap S

$$OPT(n,T) = \max\{OPT(n-1,T),\ v_n + OPT(n-1,T-w_n)\}$$



Knapsack Problem Recap

- Can solve the knapsack problem in time $O(nT)$
  - First example of dynamic programming with multiple variables in the recurrence
  - First example of a pseudopolynomial-time algorithm–compare to naïve $O(2^n)$ time algorithm
  - Later on we may see an approximation algorithm that solves knapsack in time $O(n)$ with small error

Input to a knapsack problem is $2n+1$ numbers

Single source

Shortest Paths

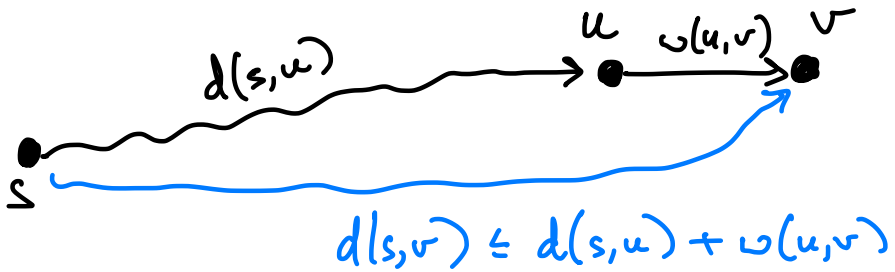
- **Input:** Directed, weighted graph $G = (V, E, \{w_e\})$, source node s
  - Possibly negative edge lengths $w_e \in \mathbb{R}$
  - No negative-length cycles!

- **Output:** Two arrays d, p
  - $d[u]$ is the length of the shortest $s \rightsquigarrow u$ path
  - $p[u]$ is the final hop on shortest $s \rightsquigarrow u$ path

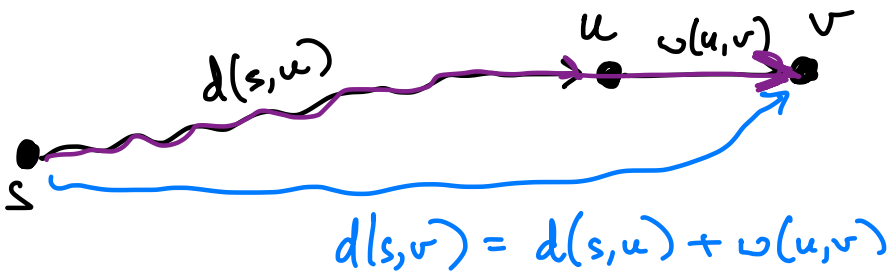
Structure of Shortest Paths

$d(u,v)$ = length of the shortest $u \rightsquigarrow v$ path

- If $(u, v) \in E$, then $d(s, v) \leq d(s, u) + w(u, v)$ for every node $s \in V$



- If $(u, v) \in E$, and $d(s, v) = d(s, u) + w(u, v)$ then there is a shortest $s \rightsquigarrow v$-path ending with (u, v)

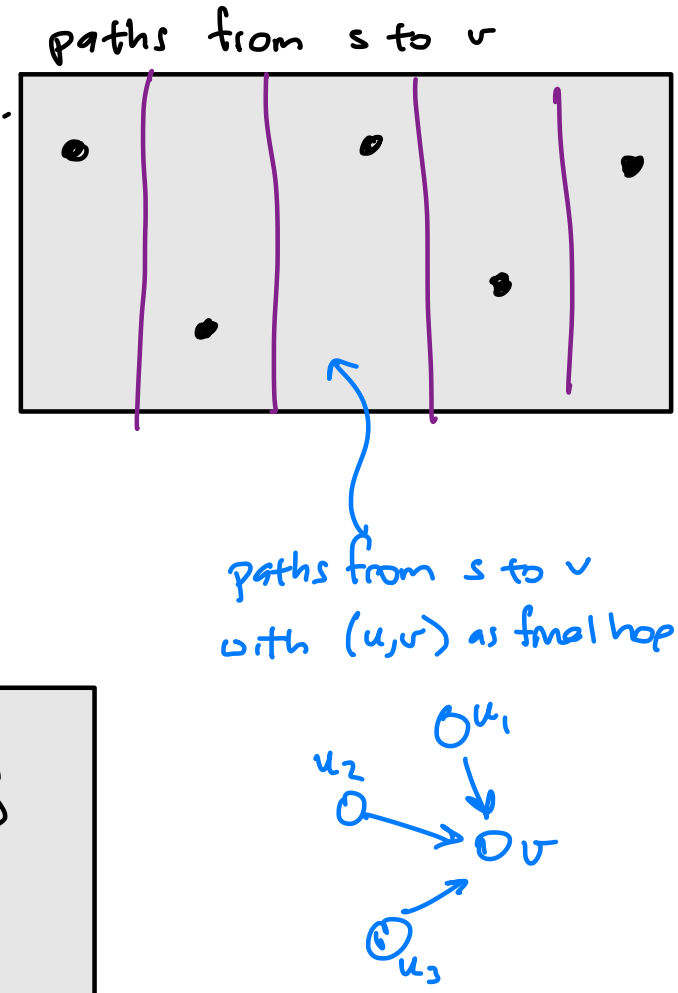


Writing the Recurrence

Subproblems

$OPT(v)$ = length of the shortest path from s to v

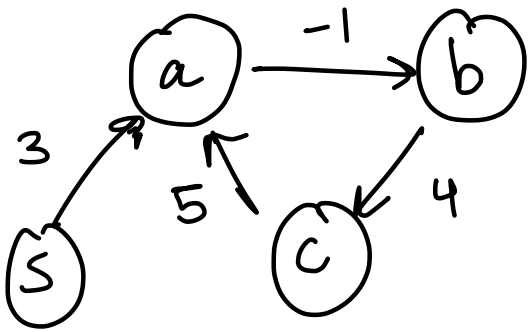
$v \in V$

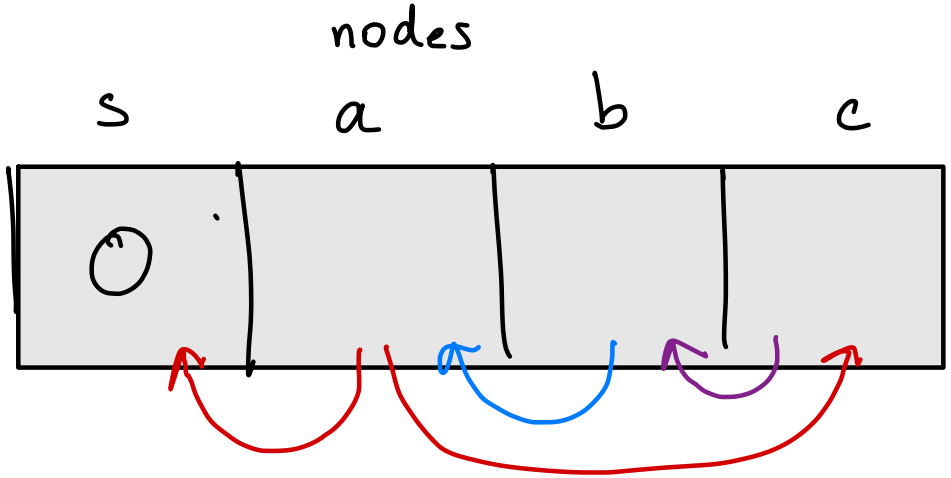


$$OPT(v) = \min_{u \text{ s.t. } (u,v) \in E} \{ OPT(u) + w(u,v) \}$$

$OPT(s) = 0$

Solving the Recurrence





Writing the Recurrence: Attempt 2

$OPT(v, i)$ = length of the shortest path from s to v using at most i edges

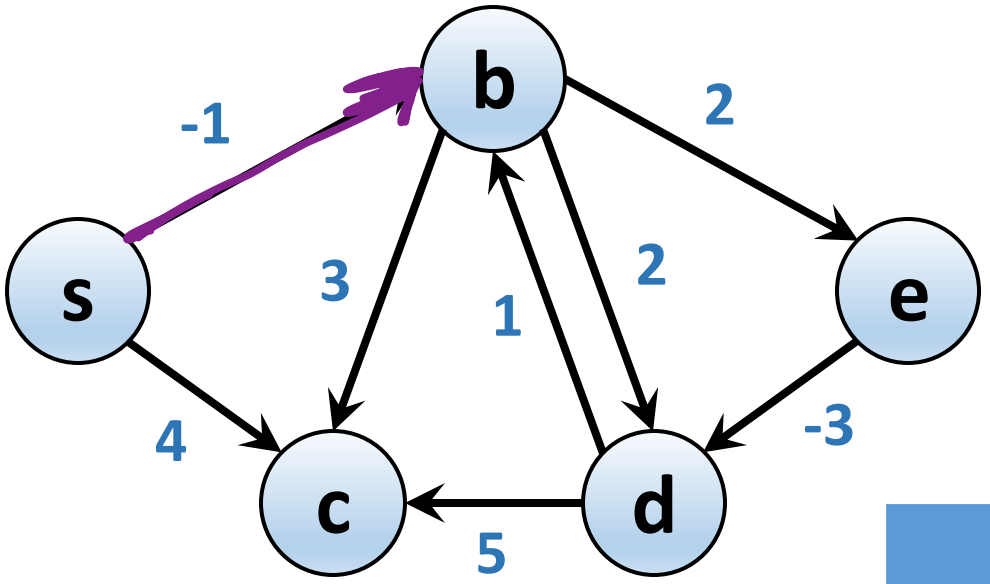
$v \in V$ $\quad i = 0, 1, \ldots, n-1$ $\quad n^2$ subproblems

$$OPT(v, i) = \min_{u : (u,v) \in E} \left\{ w(u,v) + OPT(u, i-1) \right\}$$

$OPT(s, i) = 0$ $\quad OPT(v, 0) = \infty$ for $v \neq s$

$OPT(v, n-1) = d(s, v)$ because no shortest path uses $> n-1$ hops

Solving the Recurrence: Attempt 2



$OPT(b,1) = \min\{OPT(s,0) - 1,\ OPT(d,0) + 1\}$

$= \min\{-1, \infty\} = -1$

#of hops

nodes

	0	1	2	3	4
s	0	0	0	0	0
b	∞	-1			
c	∞				
d	∞				
e	∞				

Shortest Paths Summary

- **Input:** Directed, weighted graph $G = (V, E, \{w_e\})$, and a source node s
- **Output:** Two arrays d, p
  - $d[u]$ is the length of the shortest $s \rightsquigarrow u$ path
  - $p[u]$ is the final hop on some shortest $s \rightsquigarrow u$ path

- **Negative lengths:** Bellman-Ford solves the single-source shortest paths problem in $O(nm)$ worst-case time, or finds a negative cycle
  - Often much faster in practice with suitable appropriate optimizations