CS 7800: Advanced Algorithms

Class 6: Dynamic Programming III

- Knapsack
- Shortest Paths

Jonathan Ullman
September 23, 2025

The Knapsack Problem

- **Input:** n items for your knapsack with value v_i and weight w_i and a capacity $T \in \mathbb{N}$
- **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

Solving the Recurrence

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

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

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

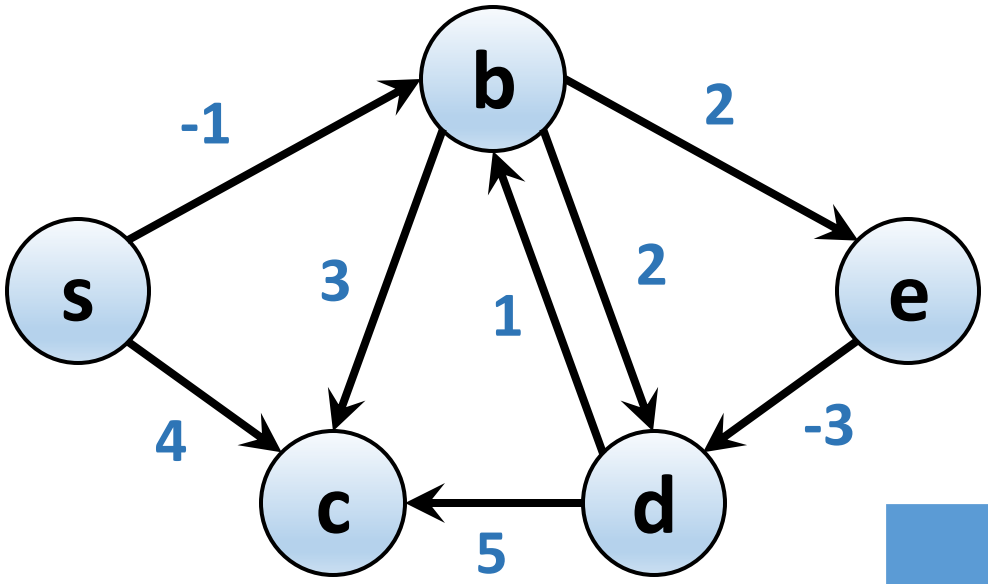
- If $(u, v) \in E$, and $d(s, v) = d(s, u) + w(u, v)$ then there is a shortest $s \rightsquigarrow v$-path ending with (u, v)

Writing the Recurrence

Solving the Recurrence

Writing the Recurrence: Attempt 2

Solving the Recurrence: Attempt 2



	0	1	2	3	4
s	0				
b	∞				
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