CS7800: Advanced Algorithms

Dynamic Programming II:

- Knapsacks
- Shortest Paths
- (Gerrymandering)

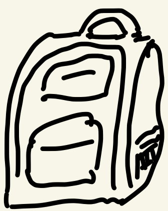
Jonathan Ullman

09-23-2022

The Knapsack Problem

Input: n items with values v_i and weights w_i
and a capacity C

Output: A subset $S \subseteq \{1, \ldots, n\}$ of items to put in
the knapsack that satisfies $\sum_{i \in S} w_i \leq C$
and has maximum value $\sum_{i \in S} v_i$



$C = 13$

$v = 8$

$w = 11$

$v = 6$

$w = 6$

$v = 3$

$w = 3$

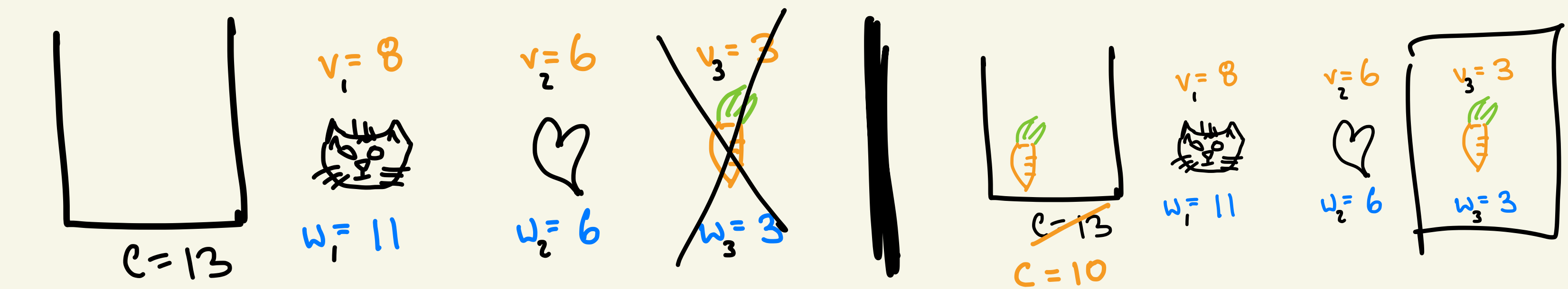
Writing a Recurrence: Take I

Let $OPT(i)$ be the optimal value using only items $1, 2, \ldots, i$ (for $i = 0, 1, \ldots, n$)

Case 1: Item i not in the optimal solution

$$OPT(i) = OPT(i-1)$$

Case 2: Item i is in the optimal solution



Writing a Recurrence: Take II

Let $OPT(i, t)$ to be the value of the optimal solution with items $1, 2, ..., i$ and capacity t

(for $i = 0, 1, ..., n$ and $t = 0, 1, ..., C$)

assume weights are integers ≥ 0

Case 1: i not in the opt solution

$$OPT(i,t) = OPT(i-1, t)$$

Case 2: i is in the optimal solution

$$OPT(i,t) = v_i + OPT(i-1, t-w_i)$$

$$\boxed{\begin{array}{l} OPT(i,t) = \max\{OPT(i-1,t),\ v_i + OPT(i-1, t-w_i)\} \\ OPT(0,t) = 0 \qquad OPT(i,0) = 0 \qquad OPT(i,-) = -\infty \end{array}}$$

Example

$S = \{3, 2\}$

$n = 3$

$C = 8$	$w_1 = 1$	$w_2 = 3$	$w_3 = 5$
	$v_1 = 4$	$v_2 = 5$	$v_3 = 8$

items \ capacities	0	1	2	3	4	5	6	7	8
3	0	4	4	5	9	9	12	12	13
2	0	4	4	5	9	9	9	9	9
1	0	4	4	4	4	4	4	4	4
0	0	0	0	0	0	0	0	0	0

Running Time Analysis

① # of subproblems = $(n+1)(C+1) = O(nC)$

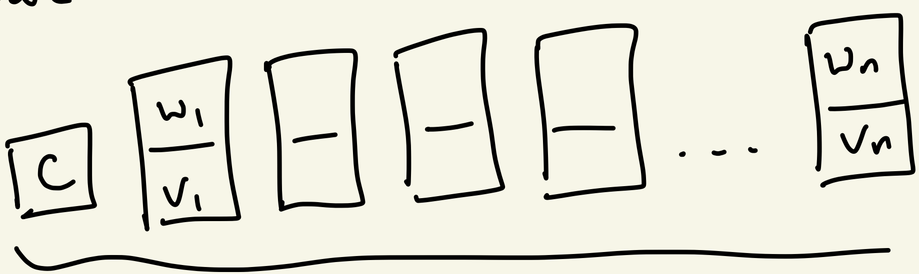
time per subproblem = $O(1)$

$O(nC)$

Compare to trying all subsets, time $O(2^n)$

② Algorithms can be hard to compare when there are multiple parameters

③



2n+1 integers

This alg is not really a poly time algorithm.

Shortest Paths

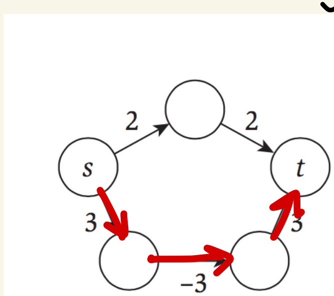
Input: Given a directed, weighted graph $G = (V, E, \{\ell_e\})$

[assume no negative length cycles]

and a pair of nodes $t \in V$

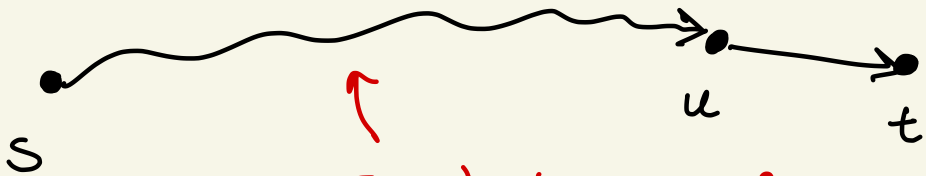
Output: A path from s to t of minimum total length $\sum_{e \in P} \ell_e$

(For every $s \in V$)



length $= 3 - 3 + 3 = 3$

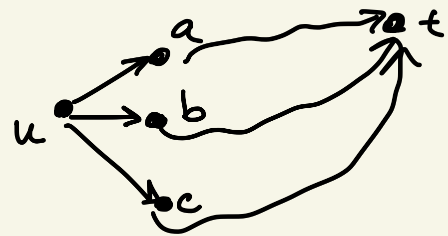
Structure of Shortest Paths



is a shortest path from s to u

Writing a Recurrence: Take I

OPT(u) is the length of the shortest path from u to t (for all $u \in V$)

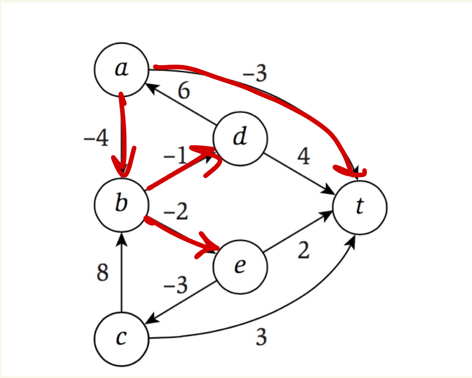


Case v: the first hop on the shortes $u \to t$ path is (u,v)

$$OPT(u) = \ell_{u \to v} + OPT(v)$$

$$\boxed{OPT(u) = \min_{v:\, u \to v \in E} \left\{ \ell_{u \to v} + OPT(v) \right\} \quad OPT(t) = 0}$$

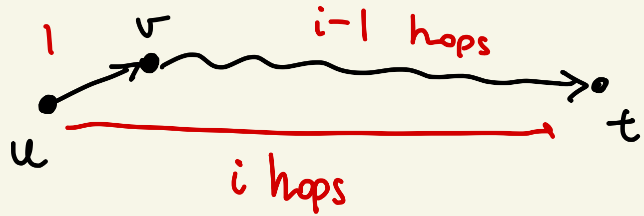
Writing a Recurrence: Take I



$$\boxed{OPT(u) = \min_{v:\, u \to v \in E} \left\{ \ell_{u \to v} + OPT(v) \right\} \qquad OPT(t) = 0}$$

node v	a	b	c	d	e	t
OPT(v)						0

Writing a Recurrence: Take II



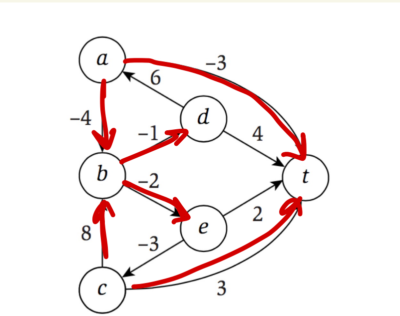
$OPT(u, i)$ to be the length of the shortest $u \to t$ path using at most i hops (for $u \in V$ and $i = 0, 1, \ldots, n-1$)

why?

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

$$OPT(t, i) = 0 \qquad \underset{u \neq t}{OPT(u, 0)} = \infty$$

Example



node v	a	b	c	d	e	t
$OPT(v, 0)$	∞	∞	∞	∞	∞	0
$OPT(v, 1)$	-3	∞	3			
$OPT(v, 2)$				3		
$\vdots$						