Learning Goals

- Describe Knapsack problem
- Develop recurrence for Knapsack
- Create dynamic programming alg
- Analyze runtime

Announcements

Knapsack Problem

Input:

Output:

Applications

Brute Force

ex: W = 5

	1	2	3	4
value	8	5	6	7
weight	2	1	3	4

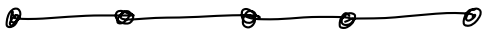
What is optimal set?

A) {1, 3} B) {1, 1} C) {1, 2, 3} D {2, 3}

Dynamic Programming

1. Think about solution as sequence of choices. Final choice?

2. For each option, think about relevant subproblem + create recurrence



$S_n = \begin{cases} \\ \\ \end{cases}$

3. Convert into recurrence for objective function

4. Pseudocode:

Knapsack Problem

Input: n items
- v_i is value of i^{th} item
- w_i is weight of i^{th} item

Capacity W (max weight)

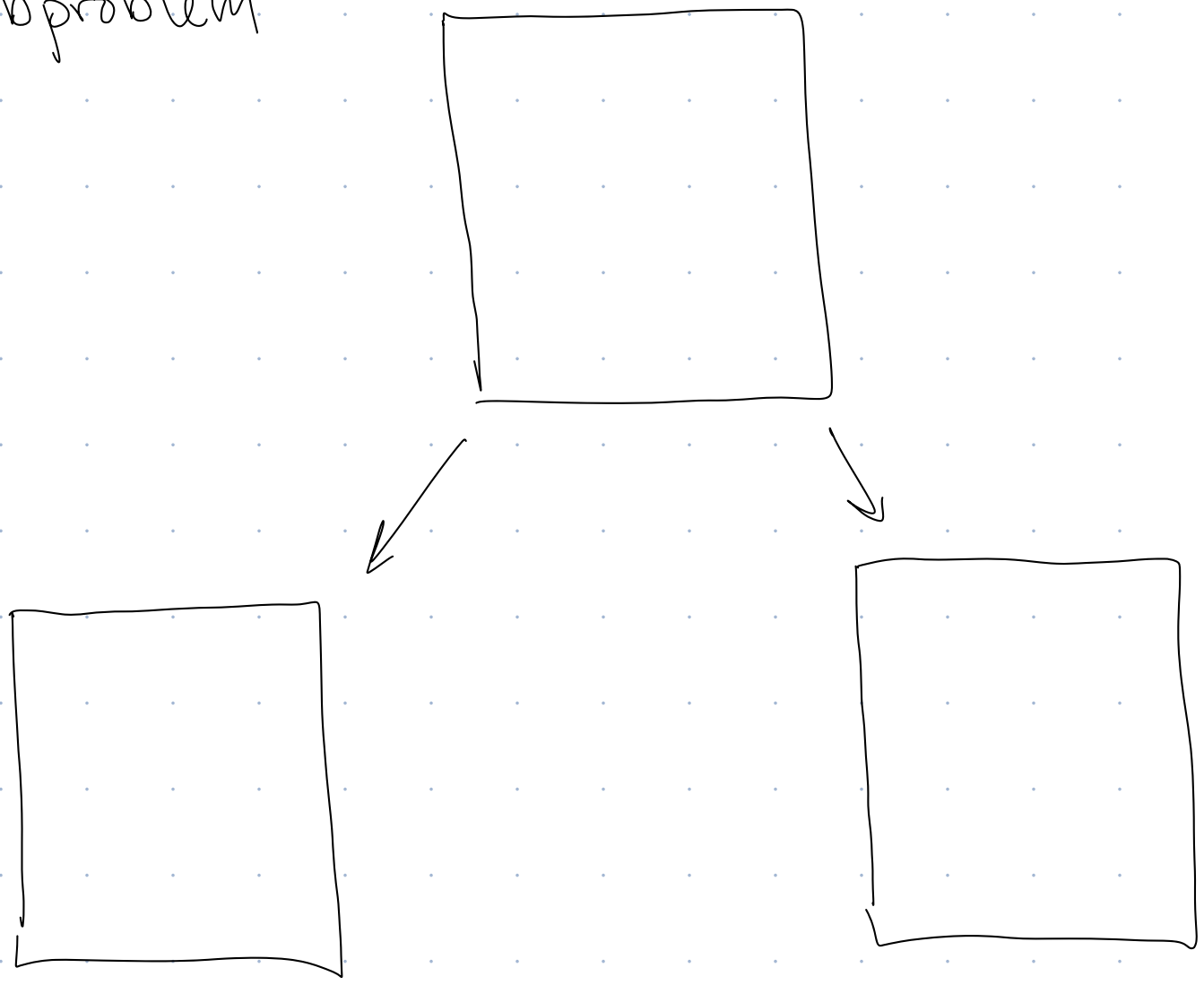
Output: $S \subseteq [n]$ ← notation $\{1, 2, 3, \ldots, n\}$

s.t. • $V(S) = \sum_{i \in S} v_i$ is maximized

• $W(S) = \sum_{i \in S} w_i \leq W$

1. Ideas for final option?

2a Subproblem



Define:

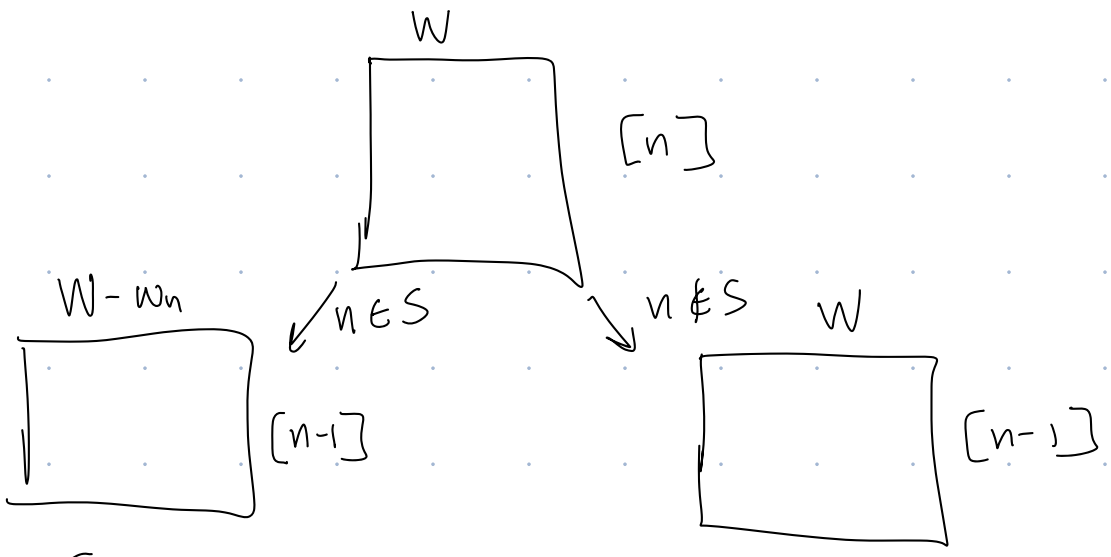
$$S_{i,w} = \begin{cases} \\ \\ \\ \end{cases}$$

	1	2	3	4
value	6	5	8	7
weight	2	1	3	4

What is $S_{2,4}$?

A) $\{1, 1\}$ B) $\{1, 2\}$ C) $\{2, 3\}$ D) $\{4\}$

2b



$$S_{n,w} = \{$$

$$V(S_{n,w}) = \{$$

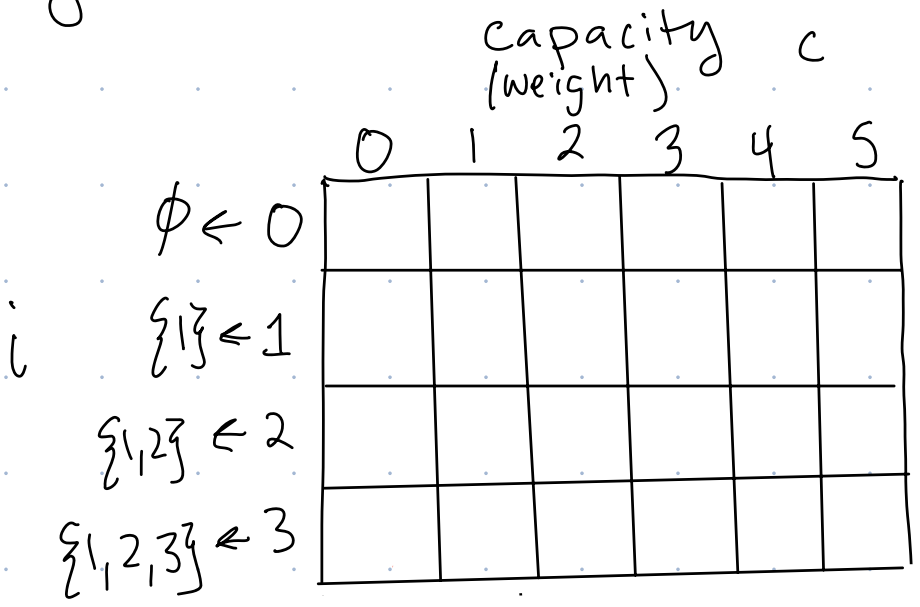
$$A[i, c] =$$

1. Fill out this array using recurrence for these items:

Input:

	1	2	3
value	6	5	8
weight	2	1	3

$W = 5$



$$A[i, c] = \begin{cases} \max\{A[i-1, c-w_i] + v_i, A[i-1, c]\} & \text{if } i \in S \text{ and } i \notin S \\ \boxed{} \end{cases}$$

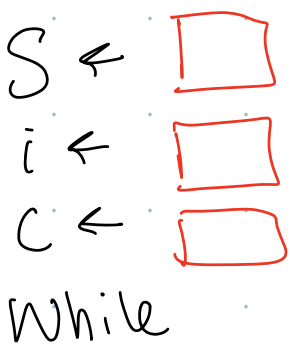
a) Determine Base Case(s)

b) Something else needed to properly use recurrence...

2. Write pseudocode to create A

Input: length-n arrays v and w, integer W

Output: Set $S \subseteq [n]$



Input: length-n arrays v and w, integer W

Output: Set $S \subseteq [n]$

Tips or working backwards through A:

Runtime