CS7800: Advanced Algorithms

Dynamic Programming I:

- Weighted Interval Scheduling
- Log cutting

Jonathan Ullman

09-20-2022

What is Dynamic Programming?

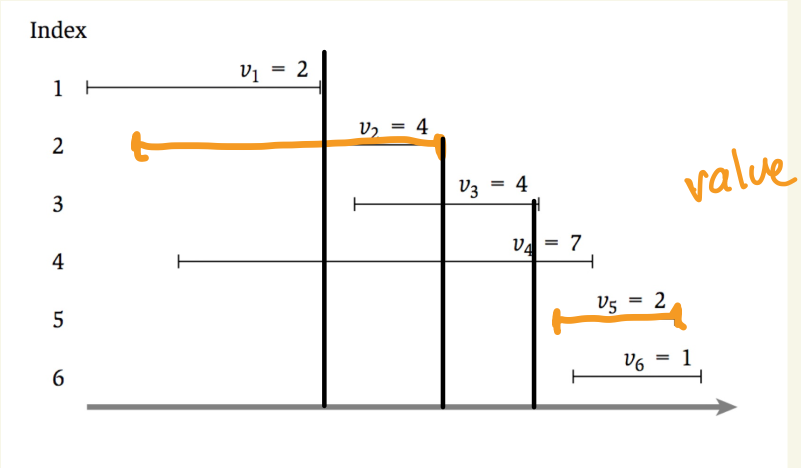
- Don't think too hard about the name
  - *I thought dynamic programming was a good name. It was something not even a congressman could object to. So I used it as an umbrella for my activities.* -Bellman

- Dynamic programming is careful recursion
  - Break the problem up into small pieces
  - Recursively solve the smaller pieces
  - **Key Challenge:** identifying the pieces

Weighted Interval Scheduling

<u>Input:</u> n intervals $[s_i, f_i]$ with values v_i

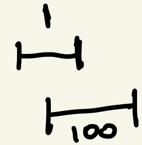
<u>Output:</u> a set of nonoverlapping intervals $S \subseteq \{1, 2, \ldots, n\}$ maximizing $\sum_{i \in S} v_i$



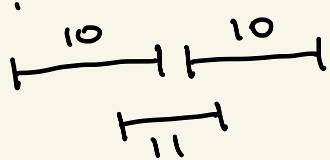
value = 4+2 = 6

A Greedy Algorithm?

- Sort by finish time?



- Sort by value?

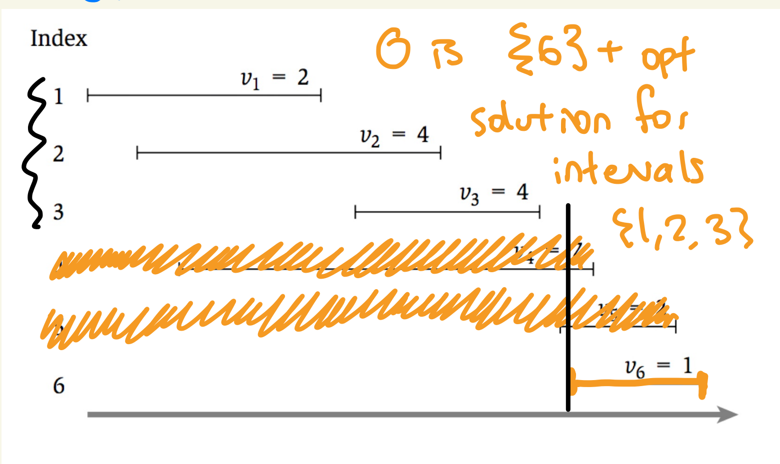


Dynamic Programming — Thinking Recursively

- Let $\mathcal{O}$ be an optimal solution

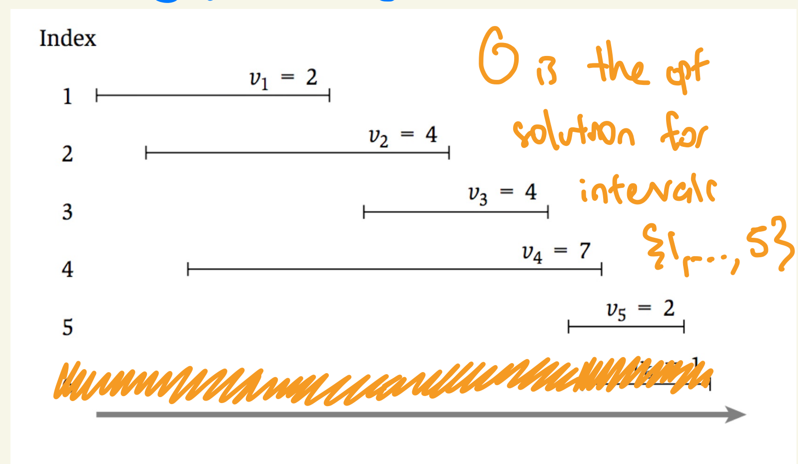
1 + value of opt for {1,2,3} = 7

Case 1: $6 \in \mathcal{O}$



0 + value of opt for {1,3,5} = 8

Case 2: $6 \notin \mathcal{O}$



Dynamic Programming – Thinking Recursively

Let $OPT(i)$ be the optimal value of a schedule using only intervals $\{1, 2, \ldots, i\}$ (for $i = 0, 1, 2, \ldots, n$)

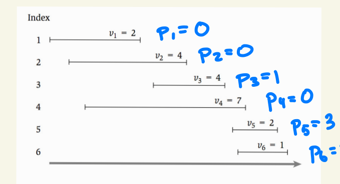
$OPT(n)$ is the thing I want

Case 1: i is in the optimal solution for $\{1, 2, \ldots, i\}$

$$OPT(i) = v_i + OPT(p_i)$$

$$(f_1 \leq f_2 \leq \ldots \leq f_n)$$

where p_i = largest j such that $f_j < s_i$

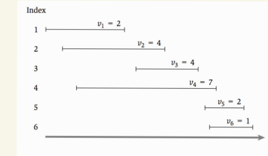


Case 2: i is not in the optimal solution

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

$$\boxed{OPT(i) = \max\{OPT(i-1),\ v_i + OPT(p_i)\} \qquad OPT(0) = 0 \qquad OPT(1) = v_1}$$

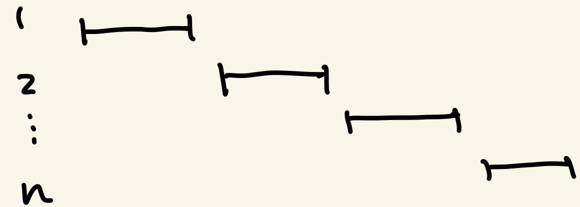
Evaluating the Recurrence: Take I

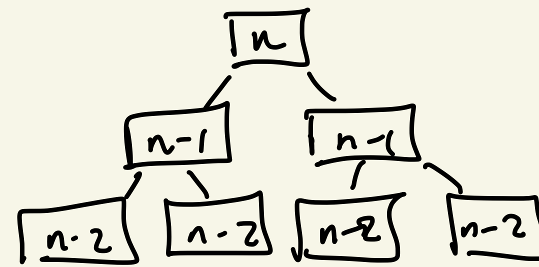


$OPT(i) = \max\{OPT(i-1),\ v_i + OPT(p_i)\}$ $\quad OPT(0) = 0 \quad OPT(1) = v_1$

```
main:
    return f(n)

f(i):
    if (i=0) return 0
    return max{f(i-1), v_i + f(p_i)}
```



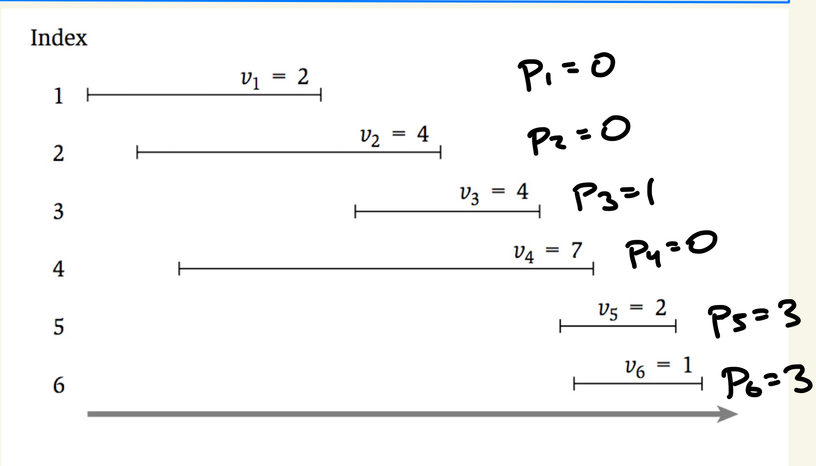


Key Observation: Dont recompute the same subproblem.

Evaluating the Recurrence: Take II

OPT(0)	OPT(1)	OPT(2)	OPT(3)	OPT(4)	OPT(5)	OPT(6)
0	2	4	6			

$OPT(i) = \max\{OPT(i-1),\ v_i + OPT(p_i)\}$ $\quad OPT(0) = 0 \quad OPT(1) = v_1$



"Bottom up":

fill in the table of subproblems one by one

Evaluating the Recurrence: Take II

OPT(0)	OPT(1)	OPT(2)	OPT(3)	OPT(4)	OPT(5)	OPT(6)
0	2	4	6			

```
main:
    opt(0) = 0
    for i = 1, ..., n:
      L opt(i) = max{ opt(i-1), v_i + opt(p_i) }
    Return opt(n)
```

running time is just $O(n)$

(once you sort the input and compute p_i)

Finding the Optimal Schedule

$$OPT(i) = \max\{OPT(i-1),\ v_i + OPT(p_i)\}$$

O_i = the better solution of $\{i\} + O_{p_i}$ and O_{i-1}

O_0	O_1	O_2	O_3	O_4	O_5	O_6
$\emptyset$	$\{1\}$	$\{2\}$	$\{1,3\}$	$\{4\}$	$\{1,3,5\}$	$\{1,3,5\}$
	yes	yes	yes	yes	yes	no

$O_6 = O_5$

$O_1 = \{1\} + \emptyset$ $\quad$ $O_3 = \{3\} + O_1$ $\quad$ $O_5 = \{5\} + O_3$

Which branch in the recurrence gives this bit.

Finding the Optimal Schedule

$$OPT(i) = \max\{OPT(i-1),\ v_i + OPT(p_i)\}$$

if this is the max then $O_i = O_{i-1}$

if this is the max then $O_i = O_{p_i} + \{i\}$

Calc OPT:

let $OPT(0) = 0$

for $i = 1, \dots, n$:

$OPT(i) = [\text{recurrence}]$

$$Branch(i) = \begin{cases} \text{yes if } v_i + OPT(p_i) \geq OPT(i-1) \\ \text{no o.w.} \end{cases}$$

Find Schedule(i)

if $i = 0$ return $\emptyset$

else if Branch(i) = yes

return $\{i\} + FS(p_i)$

else

return $FS(i-1)$

What were the steps?

requires creativity

1. We asked a good question about the solution

2. We found a recurrence relation for the value of opt

3. We implemented that recurrence efficiently
   - there is a small set of subproblems
   - avoided recomputing the same subproblem

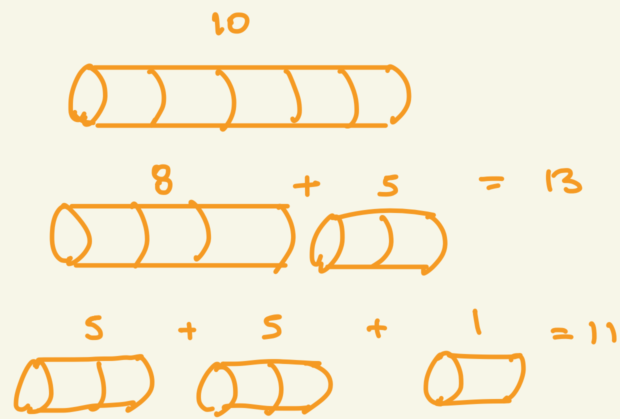
4. Add a step to find the optimal solution
   - remember which branch you took and interpret its meaning

The Log Cutting Problem

Input: The length n of a log, prices $\{p_\ell\}_{\ell=1\ldots n}$ for selling an ℓ-foot log

Output: Integer lengths $\ell_1 \ldots \ell_k$ such that $\sum_{i=1}^{k} \ell_i = n$ and $\sum_{i=1}^{k} p_{\ell_i}$ is maximized

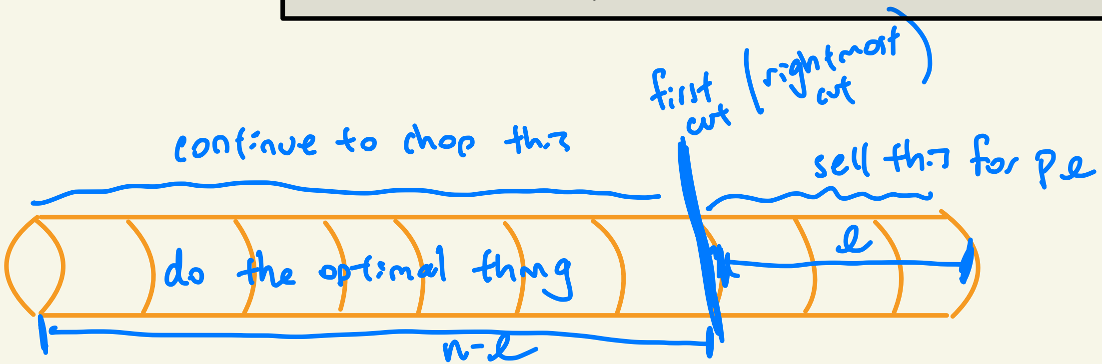
length ℓ	1	2	3	4	5
price p_ℓ	1	5	8	9	10



Dynamic Programming

Let $OPT(i)$ be the most money I can get with a log of length i (for $i = 0, 1, \dots, n$)

$$\boxed{OPT(i) = \max_{1 \le \ell \le i} \{ p_\ell + OPT(i-\ell) \} \quad OPT(0) = 0}$$



length ℓ	1	2	3	4	5	6	7	8	9	10
price p_ℓ	1	5	8	9	10	17	17	20	24	30

Calc OPT:

let $OPT(0) = 0$

for $i = 1, 2, \ldots, n$:

$\quad$ let $OPT(i) = \max\limits_{1 \le \ell \le i} \{ p_\ell + OPT(i-\ell) \}$

return $OPT(n)$

Running time is $O(n^2)$