Lesson 11. Formulating Dynamic Programming Recursions

0 Warm up

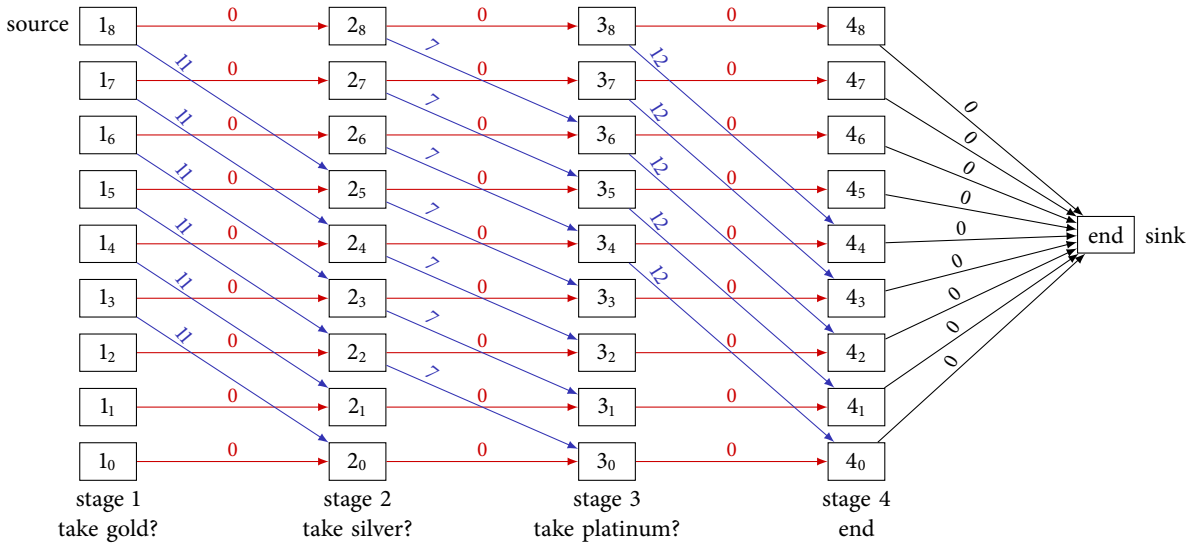
Consider the knapsack problem we studied in Lesson 5:

Example 1. You are a thief deciding which precious metals to steal from a vault:

	Metal	Weight (kg)	Value
1	Gold	3	11
2	Silver	2	7
3	Platinum	4	12

You have a knapsack that can hold at most 8kg. If you decide to take a particular metal, you must take all of it. Which items should you take to maximize the value of your theft?

- We formulated the following DP for this problem by giving the following longest path representation:



- Let $f_t(n)$ = length of a longest path from node t_n to the end node
- In the context of the knapsack problem:

$f_1(8)$ = maximum value w/ 8 kg capacity and metals 1, 2, 3 available
= optimal value of knapsack problem w/ 8 kg knapsack and metals 1, 2, 3

$f_2(5)$ = maximum value w/ 5 kg capacity and metals 2, 3 available
= optimal value of knapsack problem w/ 5 kg knapsack and metals 2, 3.

$f_3(3)$ = maximum value w/ 3kg capacity and metal 3 available
= optimal value of knapsack problem w/ 3kg knapsack and metal 3

- In other words, these are optimal values of subproblems of the knapsack problem

1 Formulating DP recursions

- Last lesson: recursions for shortest path problems
- Dynamic programs are not usually given as shortest/longest path problems
  - However, it is usually easier to think about DPs this way
- Instead, the standard way to describe a dynamic program is a recursion that defines the optimal value of one subproblem in terms of the optimal values of other subproblems
- Let's formulate the knapsack problem in Example 1 as a DP, but now by giving its recursive representation
- Let

w_t = weight of metal t $\qquad$ v_t = value of metal t $\qquad$ for $t = 1, 2, 3$

- Stages:

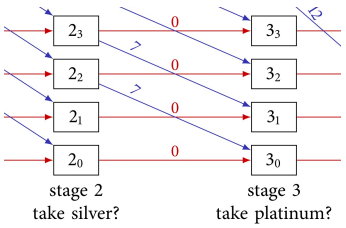
stage t $\longleftrightarrow$ { consider taking metal t $\quad t = 1, 2, 3$; end of decision-making process $\quad t = 4$

- States: state n $\longleftrightarrow$ n kg remaining in knapsack $\qquad n = 0, 1, \ldots, 8$
- Allowable decisions x_t at stage t and state n: $\longleftrightarrow$ edges outgoing from node t_n

x_t must satisfy

$x_t \in \{0, 1\}$ (0: don't take metal t; 1: take metal t)

$w_t x_t \le n$ ← we can take metal t only if we have enough capacity.



for $t = 1, 2, 3$, $n = 0, 1, \ldots, 8$

- Contribution of decision x_t at stage t and state n: $\longleftrightarrow$ edge lengths

$$v_t x_t = \begin{cases} v_t & \text{if } x_t = 1 \quad \text{(take metal } t\text{)} \\ 0 & \text{o/w} \end{cases}$$

for $t = 1, 2, 3$, $n = 0, 1, \ldots, 8$

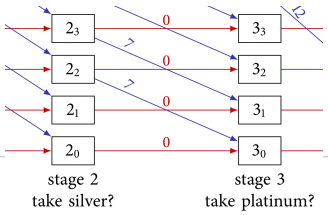
- Value-to go function $f_t(n)$ at stage t and state n:

$f_t(n)$ = maximum value w/ n kg knapsack and metals $t, t+1, \ldots, 3$ remaining. $\qquad$ for $t = 1, 2, 3, 4$, $n = 0, 1, \ldots, 8$

- Boundary conditions: $f_4(n) = 0 \qquad$ for $n = 0, 1, \ldots, 8$

- Recursion:

$$f_2(3) = \max\{0 + f_3(3),\ 7 + f_3(1)\}$$



$$f_t(n) = \max_{\substack{x_t \in \{0,1\} \\ w_t x_t \le n}} \left\{ v_t x_t + f_{t+1}(n - w_t x_t) \right\}$$

(x_t allowable)

for $t = 1, 2, 3$
$n = 0, 1, \ldots, 8$

- Desired value-to-go function value: $f_1(8)$

- In general, to formulate a DP by giving its recursive representation:

Dynamic program – recursive representation

- **Stages** $t = 1, 2, \ldots, T$ and **states** $n = 0, 1, 2, \ldots, N$
- Allowable **decisions** x_t at stage t and state n $(t = 1, \ldots, T-1;\ n = 0, 1, \ldots, N)$
- **Contribution** of decision x_t at stage t and state n $(t = 1, \ldots, T;\ n = 0, 1, \ldots, N)$
- **Value-to-go** function $f_t(n)$ at stage t and state n $(t = 1, \ldots, T;\ n = 0, 1, \ldots, N)$
- **Boundary conditions** on $f_T(n)$ at state n $(n = 0, 1, \ldots, N)$
- **Recursion** on $f_t(n)$ at stage t and state n $(t = 1, \ldots, T-1;\ n = 0, 1, \ldots, N)$

$$f_t(n) = \min_{x_t \text{ allowable}} \text{ or } \max_{x_t \text{ allowable}} \left\{ \begin{pmatrix} \text{contribution of} \\ \text{decision } x_t \end{pmatrix} + f_{t+1} \begin{pmatrix} \text{new state} \\ \text{resulting} \\ \text{from } x_t \end{pmatrix} \right\}$$

- **Desired value-to-go function value**

- How does the recursive representation relate to the shortest/longest path representation?

Shortest/longest path		Recursive
node t_n	$\leftrightarrow$	state n at stage t
edge $(t_n, (t+1)_m)$	$\leftrightarrow$	allowable decision x_t in state n at stage t that results in being in state m at stage $t+1$
length of edge $(t_n, (t+1)_m)$	$\leftrightarrow$	contribution of decision x_t in state n at stage t that results in being in state m at stage $t+1$
length of shortest/longest path from node t_n to end node	$\leftrightarrow$	value-to-go function $f_t(n)$
length of edges (T_n, end)	$\leftrightarrow$	boundary conditions $f_T(n)$
shortest or longest path	$\leftrightarrow$	recursion is min or max: $f_t(n) = \min_{x_t \text{ allowable}} \text{ or } \max_{x_t \text{ allowable}} \left\{ \begin{pmatrix} \text{contribution of} \\ \text{decision } x_t \end{pmatrix} + f_{t+1} \begin{pmatrix} \text{new state} \\ \text{resulting} \\ \text{from } x_t \end{pmatrix} \right\}$
source node 1_n	$\leftrightarrow$	desired value-to-go function value $f_1(n)$

2 Solving DP recursions

- To improve our understanding of how this recursive representation works, let's solve the DP we just wrote for the knapsack problem
- We solve the DP backwards:
  - start with the boundary conditions in stage T
  - compute values of the value-to-go function $f_t(n)$ in stages $T-1, T-2, \ldots, 3, 2$
  - …until we reach the desired value-to-go function value
- Stage 4 computations – boundary conditions:

$$f_4(n) = 0 \qquad \text{for} \quad n = 0, 1, \ldots, 8$$

- Stage 3 computations:

$$f_3(8) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 8}} \{ 12x_3 + f_4(8-4x_3) \} = \max \{ \overset{0}{0 + f_4(8)}, \overset{12}{12(1) + f_4(8-4)} \} = 12$$

$$f_3(7) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 7}} \{ 12x_3 + f_4(7-4x_3) \} = \max \{ \overset{0}{0 + f_4(7)}, \overset{12}{12(1) + f_4(7-4)} \} = 12$$

$$f_3(6) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 6}} \{ 12x_3 + f_4(6-4x_3) \} = \max \{ \overset{0}{0 + f_4(6)}, \overset{12}{12(1) + f_4(6-4)} \} = 12$$

$$f_3(5) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 5}} \{ 12x_3 + f_4(5-4x_3) \} = \max \{ \overset{0}{0 + f_4(5)}, \underset{x_3 = 1}{\overset{12}{12(1) + f_4(5-4)}} \} = 12$$

$$f_3(4) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 4}} \{ 12x_3 + f_4(4-4x_3) \} = \max \{ \overset{0}{0 + f_4(4)}, \overset{12}{12(1) + f_4(4-4)} \} = 12$$

$$f_3(3) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 3}} \{ 12x_3 + f_4(3-4x_3) \} = \max \{ \overset{0}{0 + f_4(3)} \} = 0$$

$$f_3(2) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 2}} \{ 12x_3 + f_4(2-4x_3) \} = \max \{ \overset{0}{0 + f_4(2)} \} = 0$$

$$f_3(1) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 1}} \{ 12x_3 + f_4(1-4x_3) \} = \max \{ \overset{0}{0 + f_4(1)} \} = 0$$

$$f_3(0) = \max_{\substack{x_3 \in \{0,1\} \\ 4x_3 \le 0}} \{ 12x_3 + f_4(0-4x_3) \} = \max \{ \overset{0}{0 + f_4(0)} \} = 0$$

- Stage 2 computations:

$f_2(8) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 8}} \{7x_2 + f_3(8-2x_2)\} = \max\{0 + f_3^{12}(8), 7 + f_3^{12}(6)\} = 19$

$f_2(7) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 7}} \{7x_2 + f_3(7-2x_2)\} = \max\{0 + f_3^{12}(7), 7 + f_3^{12}(5)\} = 19$

$f_2(6) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 6}} \{7x_2 + f_3(6-2x_2)\} = \max\{0 + f_3^{12}(6), 7 + f_3^{12}(4)\} = 19$

$f_2(5) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 5}} \{7x_2 + f_3(5-2x_2)\} = \max\{0 + f_3^{12}(5), 7 + f_3^{0}(3)\} = 12$ ($x_2 = 0$)

$f_2(4) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 4}} \{7x_2 + f_3(4-2x_2)\} = \max\{0 + f_3^{12}(4), 7 + f_3^{0}(2)\} = 12$

$f_2(3) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 3}} \{7x_2 + f_3(3-2x_2)\} = \max\{0 + f_3^{0}(3), 7 + f_3^{0}(1)\} = 7$

$f_2(2) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 2}} \{7x_2 + f_3(2-2x_2)\} = \max\{0 + f_3^{0}(2), 7 + f_3^{0}(0)\} = 7$

$f_2(1) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 1}} \{7x_2 + f_3(1-2x_2)\} = \max\{0 + f_3^{0}(1)\} = 0$

$f_2(0) = \max_{\substack{x_2 \in \{0,1,3\} \\ 2x_2 \le 0}} \{7x_2 + f_3(0-2x_2)\} = \max\{0 + f_3^{0}(0)\} = 0$

- Stage 1 computations – desired value-to-go function:

$f_1(8) = \max_{\substack{x_1 \in \{0,1,3\} \\ 3x_1 \le 8}} \{11x_1 + f_2(8-3x_1)\} = \max\{0 + f_2^{19}(8), 11 + f_2^{12}(5)\} = 23$ ($x_1 = 1$)

- Maximum value of theft: $f_1(8) = 23$

- Metals to take to achieve this maximum value:

$x_1 = 1, \; x_2 = 0, \; x_3 = 1 \implies$ Take metals 1 and 3