Lesson 11. Formulating Dynamic Programming Recursions

1 Formulating DP recursions

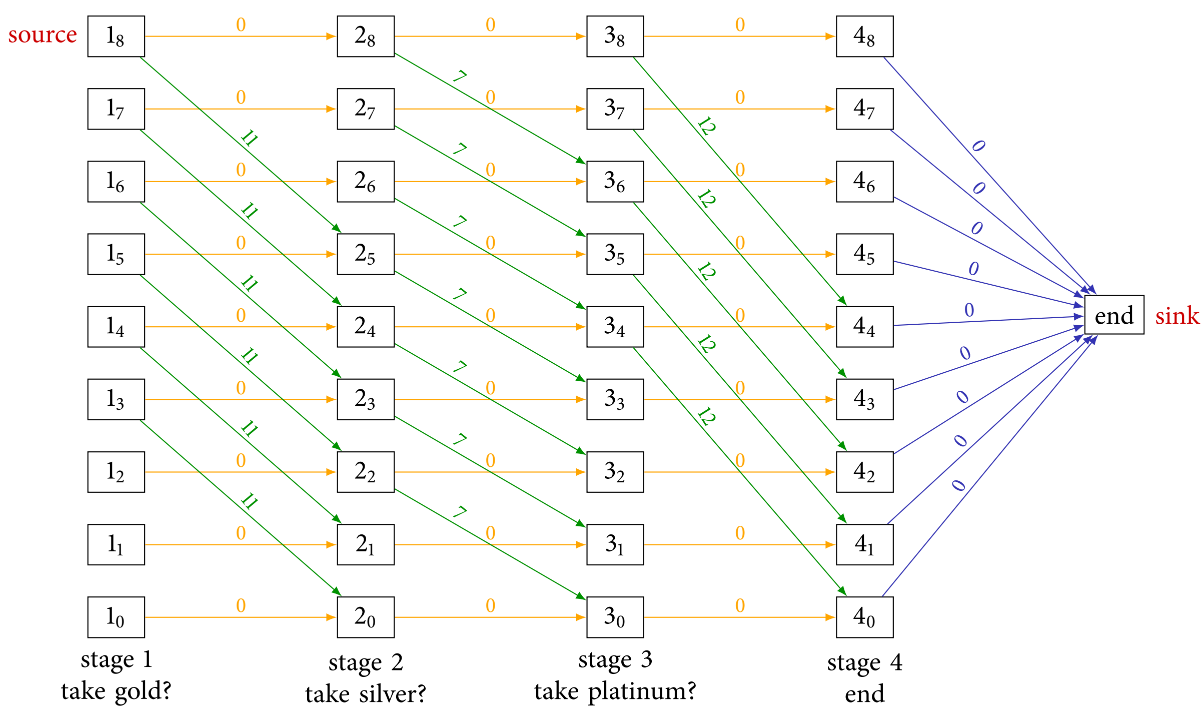
- Last lesson: recursions for shortest path problems
- Dynamic programs are not usually given as shortest/longest path problems
  - However, it is usually easier to think about DPs this way
- Instead, the standard way to describe a dynamic program is a recursion
- Let's revisit the knapsack problem that we studied back in Lesson 5 and formulate it as a DP recursion

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 dynamic program for this problem by giving the following longest path representation:



- Let's formulate this as a dynamic program, but now by giving its recursion representation

- Let

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

- Stages:

- States:

- Allowable decisions x_t at stage t and state n:

- Reward of decision x_t at stage t and state n:

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

- Boundary conditions:

- Recursion:

- Desired reward-to-go function value:

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

Dynamic program – recursive representation

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

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

- **Desired cost-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 $(t_n, (t+1)_m)$	$\leftrightarrow$	cost/reward 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	$\leftrightarrow$	cost/reward-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: $f_t(n) = \min_{x_t \text{ allowable}} \text{or} \max \left\{ \begin{pmatrix} \text{cost/reward of} \\ \text{decision } x_t \end{pmatrix} + f_{t+1} \begin{pmatrix} \text{new state} \\ \text{resulting} \\ \text{from } x_t \end{pmatrix} \right\}$
source node 1_n	$\leftrightarrow$	desired cost-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 cost-to-go function $f_t(n)$ in stages $T-1, T-2, \dots, 3, 2$
    - …until we reach the desired cost-to-go function value
- Stage 4 computations – boundary conditions:

- Stage 3 computations:

$f_3(8) =$

$f_3(7) =$

$f_3(6) =$

$f_3(5) =$

$f_3(4) =$

$f_3(3) =$

$f_3(2) =$

$f_3(1) =$

$f_3(0) =$

- Stage 2 computations:

$f_2(8) =$

$f_2(7) =$

$f_2(6) =$

$f_2(5) =$

$f_2(4) =$

$f_2(3) =$

$f_2(2) =$

$f_2(1) =$

$f_2(0) =$

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

- Maximum value of theft:

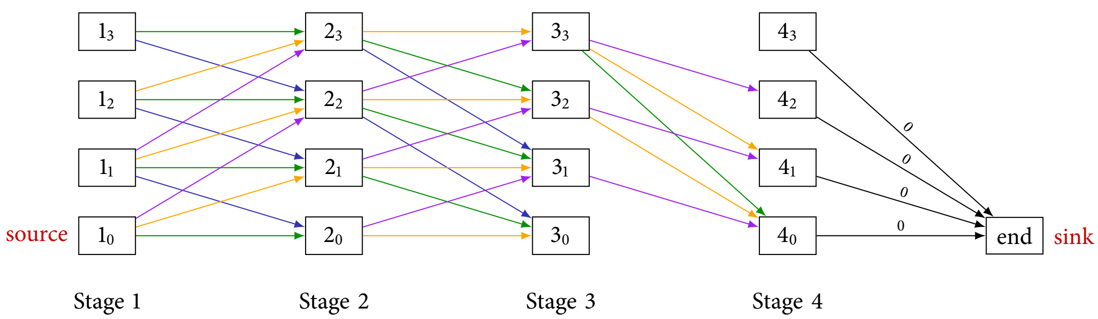
- Metals to take to achieve this maximum value:

Example 2. The Dijkstra Brewing Company is planning production of its new limited run beer, Primal Pilsner. The company must supply 1 batch next month, then 2 and 4 in successive months. Each month in which the company produces the beer requires a factory setup cost of $5,000. Each batch of beer costs $2,000 to produce. Batches can be held in inventory at a cost of $1,000 per batch per month. Capacity limitations allow a maximum of 3 batches to be produced during each month. In addition, the size of the company's warehouse restricts the ending inventory for each month to at most 3 batches. The company has no initial inventory.

The company wants to find a production plan that will meet all demands on time and minimizes its total production and holding costs over the next 3 months. Formulate this problem as a dynamic program by giving its recursive representation. Solve the dynamic program.

Formulating the DP

- Recall that in Lesson 9, we formulated this problem as a dynamic program with the following shortest path representation:
  - Stage t represents the beginning of month t ($t = 1, 2, 3$) or the end of the decision-making process ($t = 4$).
  - Node t_n represents having n batches in inventory at stage t ($n = 0, 1, 2, 3$).



Month	Production amount	Edge		Edge length
1	0	$(1_n, 2_{n-1})$	for $n = 1, 2, 3$	$1(n-1)$
1	1	$(1_n, 2_n)$	for $n = 0, 1, 2, 3, 4$	$5 + 2(1) + 1(n)$
1	2	$(1_n, 2_{n+1})$	for $n = 0, 1, 2$	$5 + 2(2) + 1(n+1)$
1	3	$(1_n, 2_{n+2})$	for $n = 0, 1$	$5 + 2(3) + 1(n+2)$
2	0	$(2_n, 3_{n-2})$	for $n = 2, 3$	$1(n-2)$
2	1	$(2_n, 3_{n-1})$	for $n = 1, 2, 3$	$5 + 2(1) + 1(n-1)$
2	2	$(2_n, 3_n)$	for $n = 0, 1, 2, 3$	$5 + 2(2) + 1(n)$
2	3	$(2_n, 3_{n+1})$	for $n = 0, 1, 2$	$5 + 2(3) + 1(n+1)$
3	0	not possible		
3	1	$(3_n, 4_{n-3})$	for $n = 3$	$5 + 2(1) + 1(n-3)$
3	2	$(3_n, 4_{n-2})$	for $n = 2, 3$	$5 + 2(2) + 1(n-2)$
3	3	$(3_n, 4_{n-1})$	for $n = 1, 2, 3$	$5 + 2(3) + 1(n-1)$

- Let d_t = number of batches required in month t, for $t = 1, 2, 3$
- Stages:

- States:

- Allowable decisions x_t at stage t and state n:

- Reward of decision x_t at stage t and state n:

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

- Boundary conditions:
- Recursion:

- Desired reward-to-go function value:

Solving the DP

- Stage 4 computations – boundary conditions:

- Stage 3 computations:

$f_3(3) =$

$f_3(2) =$

$f_3(1) =$

$f_3(0) =$

- Stage 2 computations:

$f_2(3) =$

$f_2(2) =$

$f_2(1) =$

$f_2(0) =$

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

- Minimum total production and holding cost:

- Production amounts that achieve this minimum value: