Artificial Intelligence and Machine Learning (AIML)

2023–24

AIML

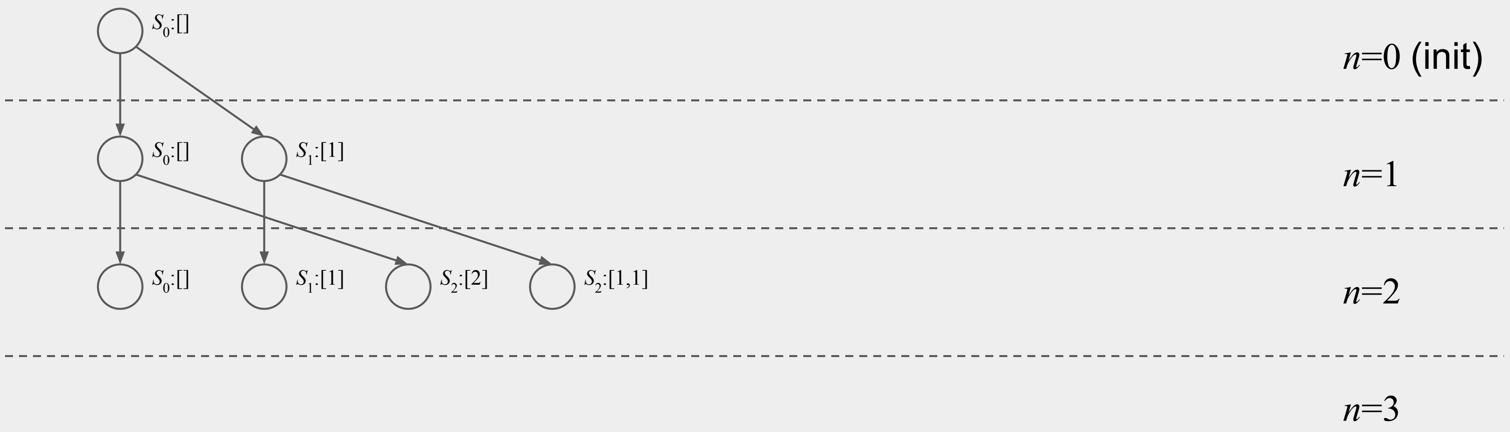
SDP: (integer) compositions

- A composition is a way of writing a whole number as a sum of whole numbers, where the order of the terms in the sum matters
- **Example**: 4 has eight compositions [4], [1, 3], [2, 2], [1, 1, 2], [3, 1], [1, 2, 1], [2, 1, 1], [1, 1, 1, 1] where e.g. [1, 3] means 1+3, [3, 1] means 3+1 and [1, 1, 2] means 1+1+2
- A simple n-step procedure for generating all compositions, can be described as follows: iterate through integers n, and on each stage, keep the current composition, or extend with a single value which makes it add up to n
- For computational convenience we use a **memoized SDP** to keep track of the stage (S_0, S_1, S_2 and so on) in which the configuration was generated

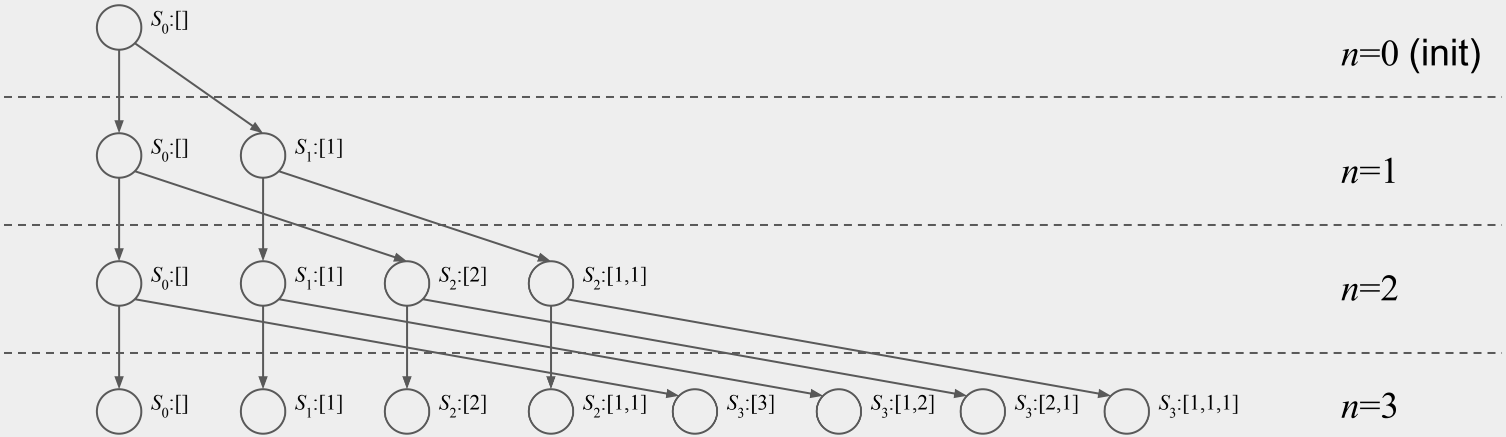
SDP: exhaustive compositions

SDP: exhaustive compositions

SDP: exhaustive compositions



SDP: exhaustive compositions



DP: optimal plant cutting

- Solve the **maximum sum composition** optimization problem:

$$X^{\star} = \arg\max_{X' \in \mathcal{X}} \left(\sum_{i \in X} \Pr(i) \right)$$

where $\mathcal{X}$ is the set of all compositions of n, mapped to the data $\Pr$.

- In Tutorial 5 problem Q1, an element of a composition is a plant cut length (units), and the data $\Pr$ is the price of each length:

Length (units)	Price (£)
1	2
2	4
3	7
4	3
5	9

DP: optimal plant cutting

- Corresponding DP Bellman recursion:

$$P_0^\star = 0$$
$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i) \right)$$

- Maximum is over all previously retained (memoized) stages (compositions of i)
- The best price for length 0 plant is zero. Otherwise, the best price of a plant of length n, is the best composition of plant of length i extended to length n (**SDP: extension**, $\Pr(i)$ term) using the best (cut) plant at length $n{-}i$ (**SDP: keep current composition,** P^*_{n-i} term)
- Computational graph illustration (feasible up to length n=4).

DP: optimal plant cutting

$$P_0^\star = 0$$

$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i) \right)$$

S_0:[]

$P^*_0 = 0$, $X^*_0 = [\ ]$

n=0 (init)

n=1

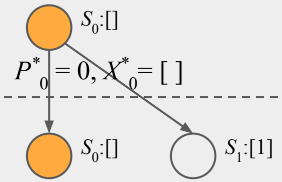
n=2

n=3

n=4

DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i)\right)$$



n=0 (init)

n=1

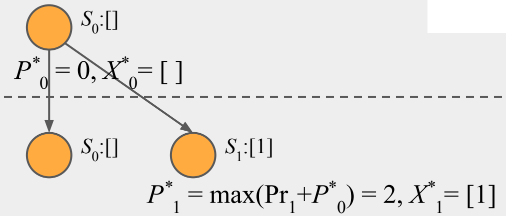
n=2

n=3

n=4

DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i)\right)$$



n=0 (init)

n=1

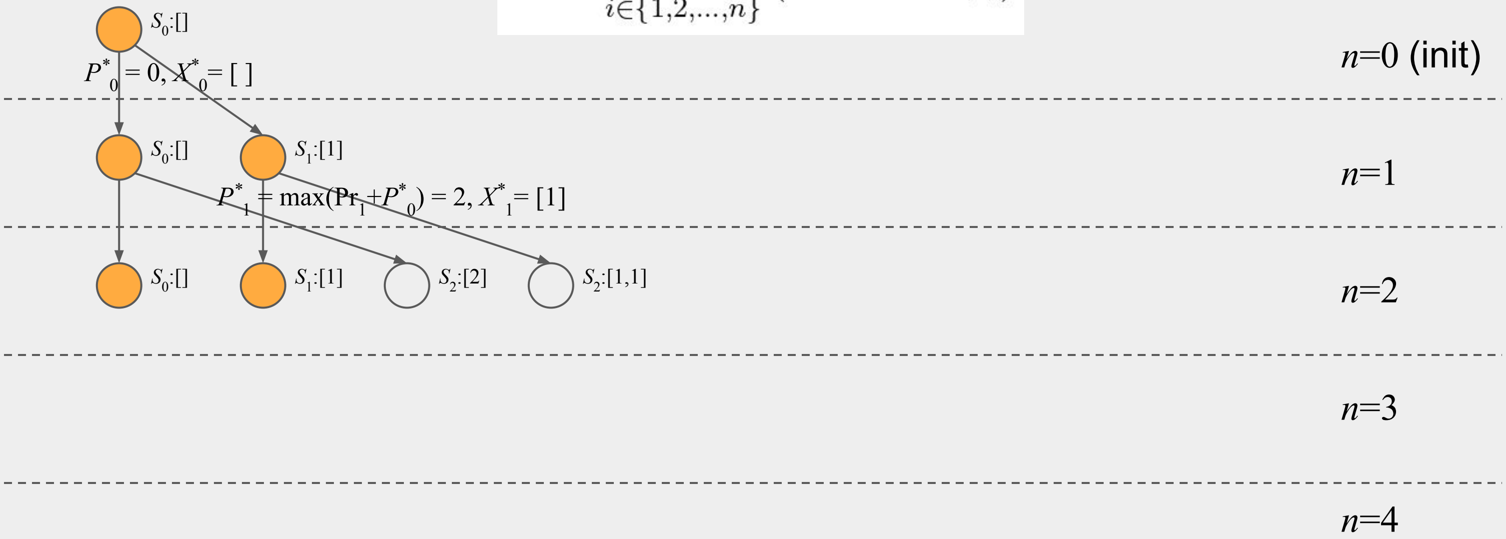
n=2

n=3

n=4

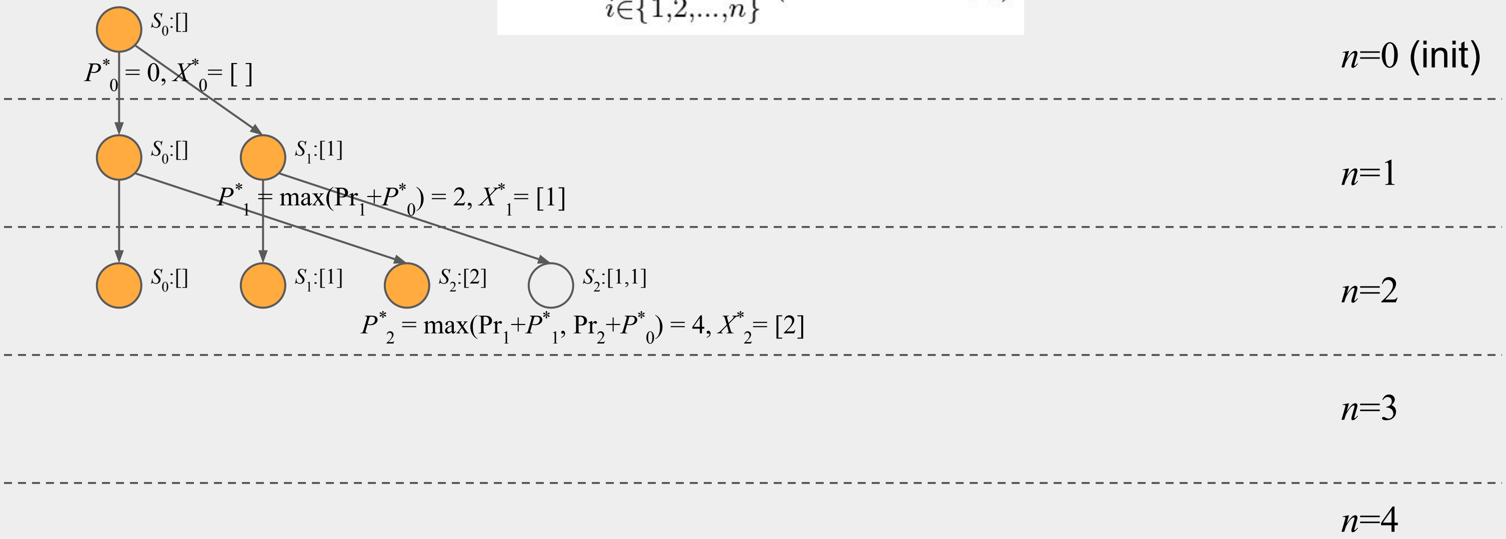
DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i)\right)$$



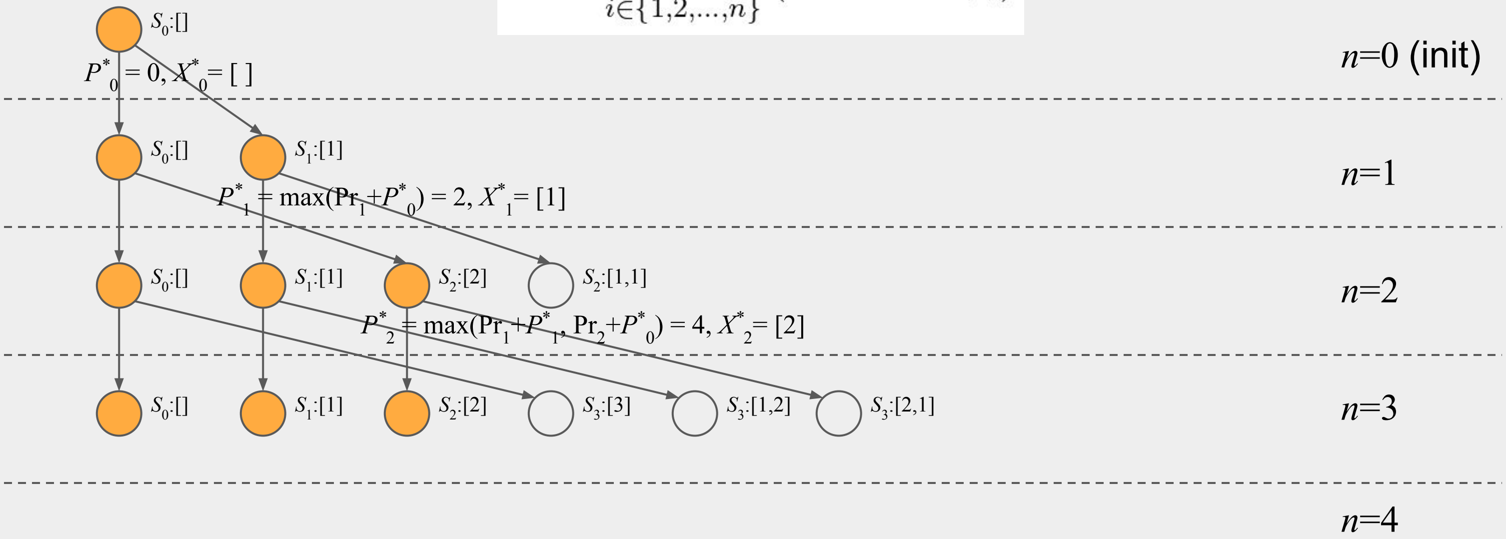
DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \mathrm{Pr}\,(i)\right)$$



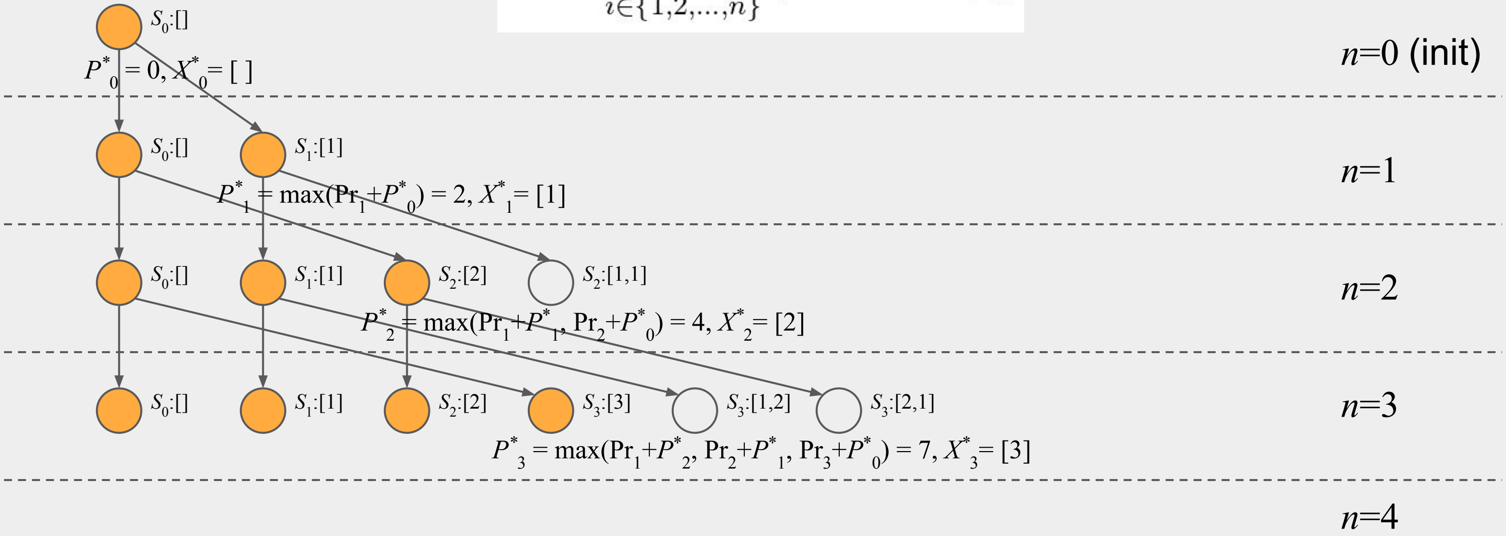
DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i\in\{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i)\right)$$



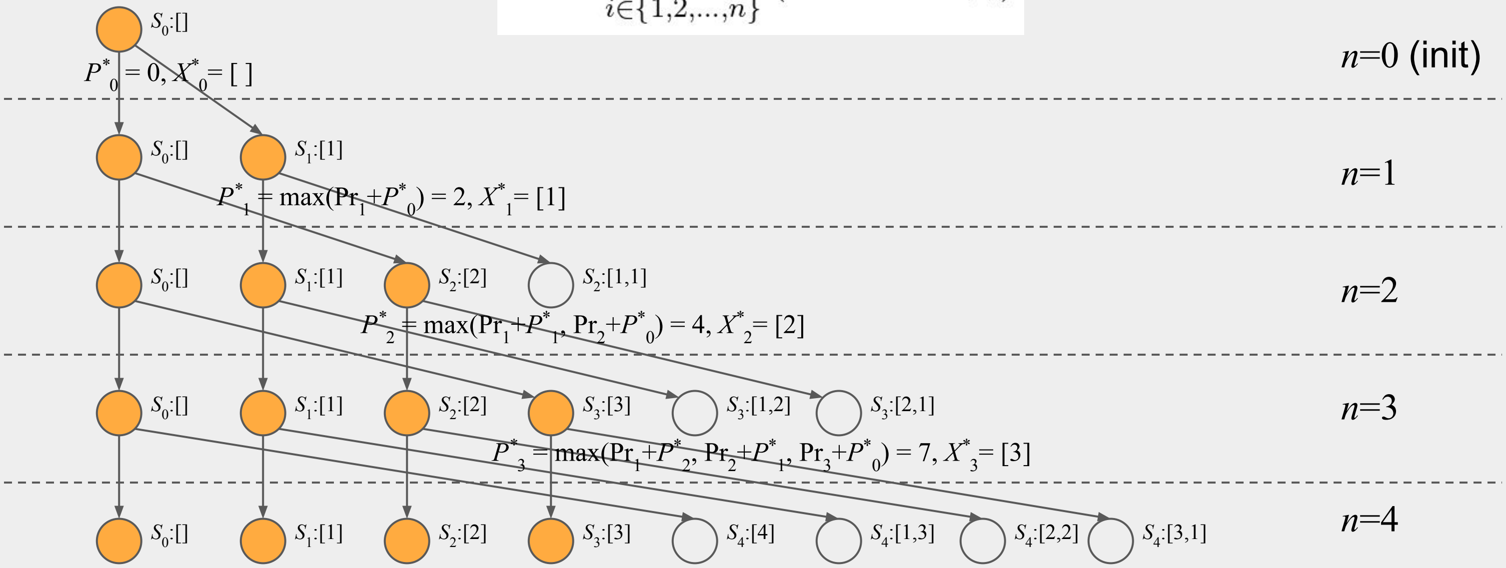
DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i)\right)$$



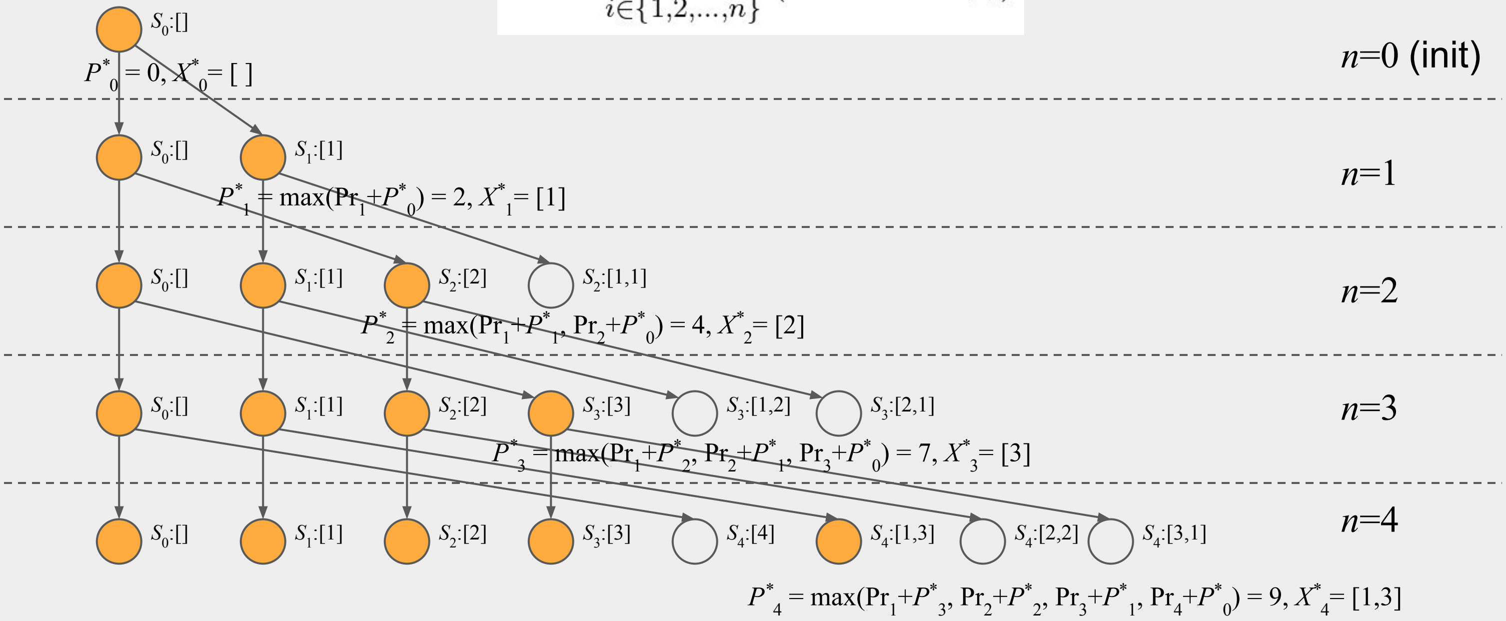
DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i \in \{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i)\right)$$



DP: optimal plant cutting

$$P_0^\star = 0$$
$$P_n^\star = \max_{i\in\{1,2,\ldots,n\}} \left(P_{n-i}^\star + \Pr(i)\right)$$



DP: composition algorithm analysis

- Solving the problem exhaustively requires generating all 2^{n-1} compositions of n: exponential time (and space) complexity, $O(2^n)$
- DP solution compares up to n previous compositions over n iterations, taking n^2 time: polynomial time complexity $O(n^2)$
- DP solution memory required, must store (memoize) n previous optimal solutions: linear space complexity, $O(n)$