MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

EECS 376 Discussion

Daphne Tsai

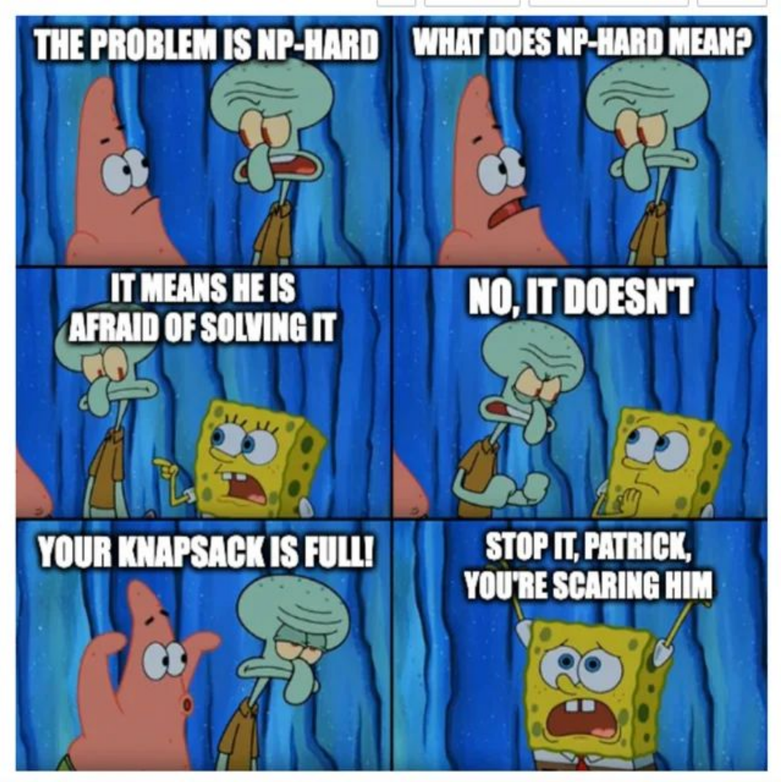
Week 3: Dynamic Programming

9/15/23

Friday 11:30am @ NAME 138

Today + announcements

- Dynamic Programming :)
- Next two weeks: CSRB 2246



Dynamic Programming

- We don't want to solve the same problem multiple times!
- Is there a way to store previous results? → memo

- WHY do we use a memo?

When do I use Dynamic Programming?

- Optimal substructure
- Overlapping subproblems (ex. Fibonacci numbers)

How do I approach Dynamic Programming?

1. Write the recurrence relation
   a. How do I split this problem up?
2. Bound the number of distinct subproblems
   a. How many variables in each subproblem?
3. How should I store the results?
   a. Bottom up
   b. Top down (Recursion)
   c. Top down (Memo)

Slide modified from Aditya Singhvi

The 0-1 Knapsack Problem

The 0-1 Knapsack Problem Formally

- You have a set of n items, each with weight w_i and value v_i. Additionally, you have a knapsack with maximum weight capacity C.

- Inputs:
  - n-length array of positive integer weights $W = (w_1, w_2, \dots, w_n)$
  - n-length array of positive integer values V= $(v_1, v_2, \dots, v_n)$
  - Capacity of the knapsack $C \in \mathbb{N}$

- Goal: pick a subset of items $S \subseteq \{1,2, \dots n\}$ that maximizes the value of the knapsack $\sum_{i \in S} v_i$, while staying within the capacity $\sum_{i \in S} w_i \leq C$.

Slide modified from Julie Shah

How do we solve the Knapsack Problem?

$$K(i, C) = \max\{K(i-1, C-w_i) + v_i, K(i-1, C)\}$$

Given an array of $n \geq 1$ *positive real numbers* (represented as constant size floating points), $A[1..n]$, we are interested in finding the smallest *product* of any subarray of A, i.e.,

$$\min\{A[i]A[i+1]\cdots A[j] : i \leq j \text{ are indices of } A\}.$$

Give a recurrence relation, then a DP solution in O(n) time

3	0.8	5	0.6	0.5

We have a collection M of chicken McNuggets meals; these meals are displayed to you in a menu, represented as an array $M[1..n]$, with the number of McNuggets per meal. Your goal is to determine, for a given positive integer t, whether it is possible to consume *exactly* t McNuggets *using at most one instance of each meal*[1]. For example, for $M = [1, 2, 5, 5]$ and $t = 8$, it is possible with $M[1] + M[2] + M[3] = 8$; however, for the same M and $t = 4$, it is not possible.

Give a recurrence relation