CS 141: Intermediate Data Structures and Algorithms

Discussion - Week 6, Winter 2018

Dynamic Programming

- General idea
- Examples

General idea

- Applicable when subproblems are not independent : Subproblems share sub-subproblems.
- A divide and conquer approach would repeatedly solve the common subproblems
- Dynamic programming solves every subproblem just once and stores the answer in a table.

Elements of Dynamic Programming

- Optimal Substructure
  - An optimal solution to a problem contains within it an optimal solution to subproblems
  - Optimal solution to the entire problem is build in a bottom-up manner from optimal solutions to subproblems
- Overlapping Subproblems
  - If a recursive algorithm revisits the same subproblems over and over ⇒ the problem has overlapping subproblems

Dynamic Programming Algorithm

1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution in a bottom-up fashion
4. Construct an optimal solution from computed information

Example 1: Find *nth* Fibonacci number

0 1 1 . . . $F(n\text{-}2)$ $F(n\text{-}1)$ $F(n)$

F(0) = 0

F(1) = 1

F(2) = 1+0 = 1

…

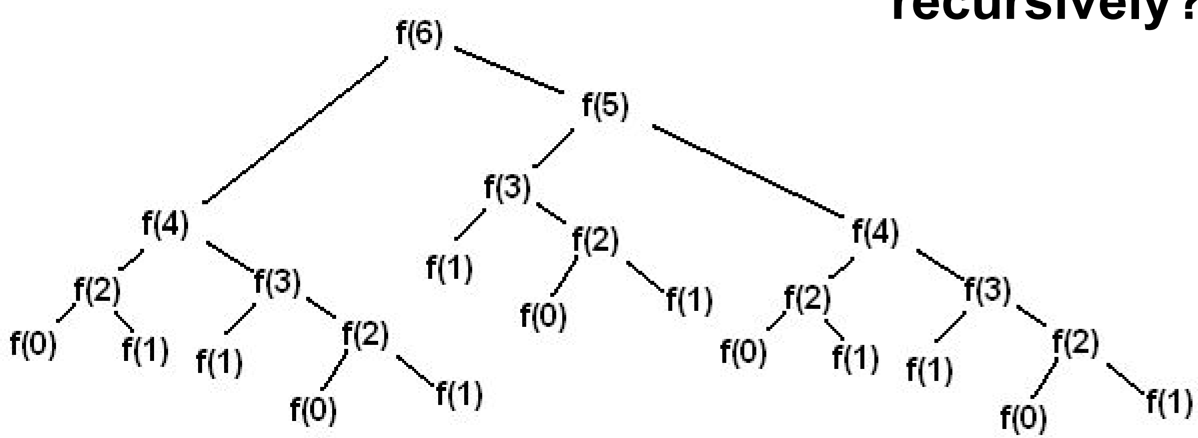
F(n-2) =

F(n-1) =

F(n) = F(n-1) + F(n-2)

Efficiency in time?

What if we solve it recursively?



Example 2: Matrix chain multiplication

$A_1 \cdot A_2 \cdot A_3$

- A_1: 10 x 100
- A_2: 100 x 5
- A_3: 5 x 50

1. $((A_1 \cdot A_2) \cdot A_3)$: $A_1 \cdot A_2$ = 10 x 100 x 5 = 5,000 (10 x 5)

   $((A_1 \cdot A_2) \cdot A_3)$ = 10 x 5 x 50 = 2,500

   Total: 7,500 scalar multiplications

2. $(A_1 \cdot (A_2 \cdot A_3))$: $A_2 \cdot A_3$ = 100 x 5 x 50 = 25,000 (100 x 50)

   $(A_1 \cdot (A_2 \cdot A_3))$ = 10 x 100 x 50 = 50,000

   Total: 75,000 scalar multiplications

Example 2: Matrix chain multiplication

$m[i, j] = 0$ if $i = j$

$m[i, j] = \min_{i \le k < j} \{m[i, k] + m[k+1, j] + p_{i-1}p_kp_j\}$ if $i < j$

Example 2: Group activity

$A_1A_2A_3A_4$

A_1: 10x5

A_2: 5x20

A_3: 20x4

A_4: 4x25