Chapter 5, Dynamic Programming:

- Introduction and sample problems (**Today**)
- Principles of DP (next week Monday)
- DP algorithms, solutions to sample problems (next week Wednesday)
- Optimal BSTs (week-after Monday)
- Finish optimal BSTs/review for test (week-after Wednesday)
- [Test 2, Thur, Apr 4, *not* covering DP]

Today:

- Lab follow-up
- Goals of the unit
- Overlapping subproblems
- The coin-changing problem
- Three sample problems

What dynamic programming is:

> *An algorithmic technique for efficiently solving an optimization problem with overlapping subproblems by storing the results of subproblems in a table.*

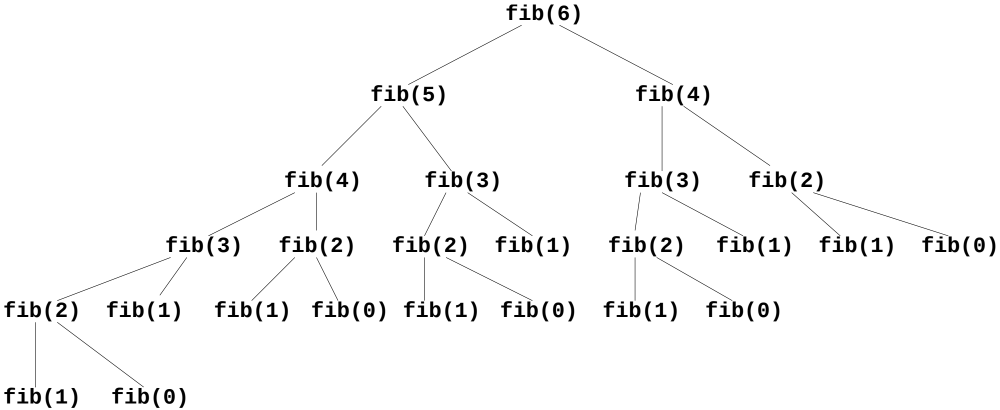
Our goals for dynamic programming in CSCI 345:

- Know what dynamic programming is and what kind of problems it applies to.
- Understand the principles of dynamic programming and the terminology used to talk about it.
- Be able to take a problem *and its recursive characterization* (the mathematical formulation of its solution) and code up an algorithm to compute the maximum value or minimum cost.

Not goals in CSCI 345 (come back for DP unit in CSCI 445):

- Be able to take a problem and devise a recursive characterization.
- Having devised a recursive characterization, be able to code up an algorithm to compute the maximum value or minimum cost *and to reconstruct the optimal solution*.

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$



Recursive characterization. A formula relating problems to subproblems of the same kind.

Overlapping subproblems. The situation when the recursion tree for a formula contains multiple instances of the same subproblem.

Memoization. Storing the results of subproblems for later retrieval.

Top-down approach. Beginning the computation of a recursive formula from the top-level problem, computing subproblems on-demand.

Bottom-up approach. Beginning the computation of a recursive formula from the base cases and building the results of other subproblems from there.

Given an amount a and a list of coin denominations D, find a list of coin quantities L such that

$$\sum_{i=0}^{n-1} D[i]L[i] = a$$

and that minimizes $\sum_{i=0}^{n-1} L[i]$

The minimum *number of coins* is

$$\min_{L} \sum_{i=0}^{n-1} L[i]$$

The bag of coins that minimizes that number is

$$\arg\min_{L} \sum_{i=0}^{n-1} L[i]$$

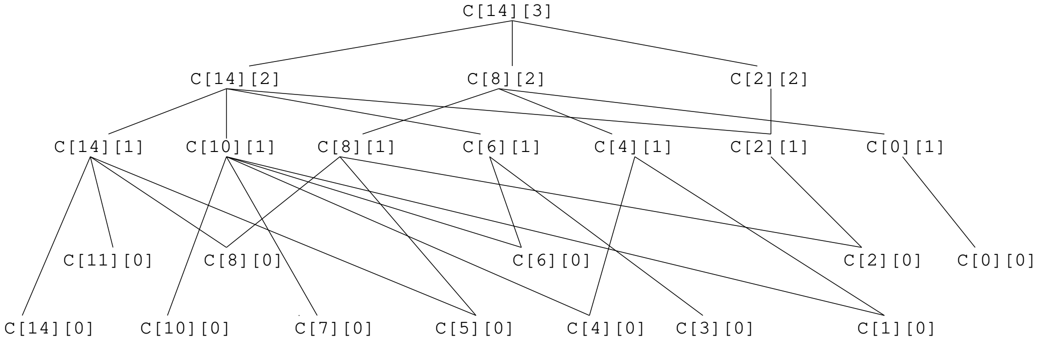
Imagine a system with coins $[1, 3, 4, 6]$ and making change for for 14. Let $C[i][j]$ stand for the fewest number of coins needed to make change for amount *i using only coins 0 through j*.

$$C[14][3] = \min \begin{cases} 0 + C[14][2] & \text{no hexes plus best change for 14 with remaining coins} \\ 1 + C[8][2] & \text{one hex plus best change for 8 with remaining coins} \\ 2 + C[2][2] & \text{two hexes plus best change for 2 with remaining coins} \end{cases}$$

$$C[14][3] = \min \begin{cases} 0 + C[14][2] & = & 0 + 4 & = & 4 \\ 1 + C[8][2] & = & 1 + 2 & = & 3 \\ 2 + C[2][2] & = & 2 + 2 & = & 4 \end{cases}$$

Let $C[i][j]$ stand for the fewest number of coins needed to make change for amount i *using only coins 0 through* j.

$$C[i][j] = \begin{cases} 0 & \text{if } i = 0 \\ i & \text{if } j = 0 \\ \min_{0 \leq k < \frac{i}{D[j]}} \{k + C[i - k \cdot D[j]][j-1]\} & \text{otherwise} \end{cases}$$


C[14][3]
C[14][2]
C[8][2]
C[2][2]
C[14][1]
C[10][1]
C[8][1]
C[6][1]
C[4][1]
C[2][1]
C[0][1]
C[11][0]
C[8][0]
C[6][0]
C[2][0]
C[0][0]
C[14][0]
C[10][0]
C[7][0]
C[5][0]
C[4][0]
C[3][0]
C[1][0]

0-1 Knapsack.

Given a capacity c and the value and weight of n items in arrays V and W, find a subset of the n items whose total weight is less than or equal to the capacity and whose total value is maximal.

	0	1	2	3
V	20	15	90	100
W	1	2	4	5

$c = 7$

set	weight	value	
$\{2, 3\}$	9	190	*exceeds capacity*
$\{1, 3\}$	7	115	*not optimal*
$\{0, 1, 2\}$	7	125	*optimal*

Longest common subsequence.

Given two sequences, find the longest subsequence that they have in common.

D	A	T	A	S	T	R	U	C	T	U	R	E	S
A	L	G	O	R	I	T	M	S					

A A A A A B / A B A A A A not A A A A A B / A B A A A A

A A A A A B A A A A / A B A A A A not A A A A A B A A A A / A B A A A A

Matrix multiplication.

> *Given $n+1$ dimensions of of n matrices to be multiplied, find the optimal order in which to multiply the matrices, that is, find the parenthesization of the matrices that will minimize the number of scalar multiplications.*

Assume the following matrices and dimensions: $A, 3 \times 5$; $B, 5 \times 10$; $C, 10 \times 2$, $D, 2 \times 3$; $E, 3 \times 4$.

$$
\begin{array}{lrcl}
(A \times B) \times (C \times (D \times E)) & 3 \cdot 5 \cdot 10 + 2 \cdot 3 \cdot 4 + 10 \cdot 2 \cdot 4 + 3 \cdot 10 \cdot 4 & = & 374 \\
(A \times (B \times C)) \times (D \times E) & 5 \cdot 10 \cdot 2 + 2 \cdot 3 \cdot 4 + 3 \cdot 5 \cdot 2 + 3 \cdot 2 \cdot 4 & = & 178 \\
A \times (B \times (C \times (D \times E))) & 2 \cdot 3 \cdot 4 + 10 \cdot 2 \cdot 4 + 5 \cdot 10 \cdot 4 + 3 \cdot 5 \cdot 4 & = & 364
\end{array}
$$

Problem	*Thing to find*	*Optimization*	*Constraint*
Coin-changing	A set of coins.	Minimize the number of coins.	The coins' values sum to the given amount.
Knapsack	A set of objects	Maximize the sum of the objects' values.	The sum of the objects' weights doesn't exceed the given capacity.
Longest common subsequence	A subsequence in each of two given sequences.	Maximize the length of the subsequences.	The subsequences have the same content.
Matrix multiplication	A way to parenthesize the the matrices being multiplied.	Minimize the number of scalar multiplications required.	The parenthesization is complete and mathematically coherent.
Optimal BST	A BST for a given set of keys	Minimize the expected length of a search.	The tree satisfies the criteria for a BST.

Coming up:

Do **Traditional RB** *project (due Fri, Mar 22)*
(Recommended: Do **LL RB** *project for your own practice)*

Due **Fri, Mar 22—today** *(end of day)*
Take DP intro quiz

Due **Mon, Mar 26** *(class time)*
Read Section 6.(1&2)
Do Exercises 6.(5–7)

Due **Tues, Mar 26** *(end of day)*
Read Section 6.3
Do Exercises 6.(16, 19, 23, 33)
Take quiz (DP principles)

Due **Wed, Mar 27** *(class time)*
Read Section 6.4
Do Exercises 6.(20, 24, 32)

Due **Thurs, Mar 28** *(end of day)*
Take quiz (DP algorithms)