Data Structures and Algorithm Analysis (CSC317)

Dynamic Programming 1

Odelia Schwartz

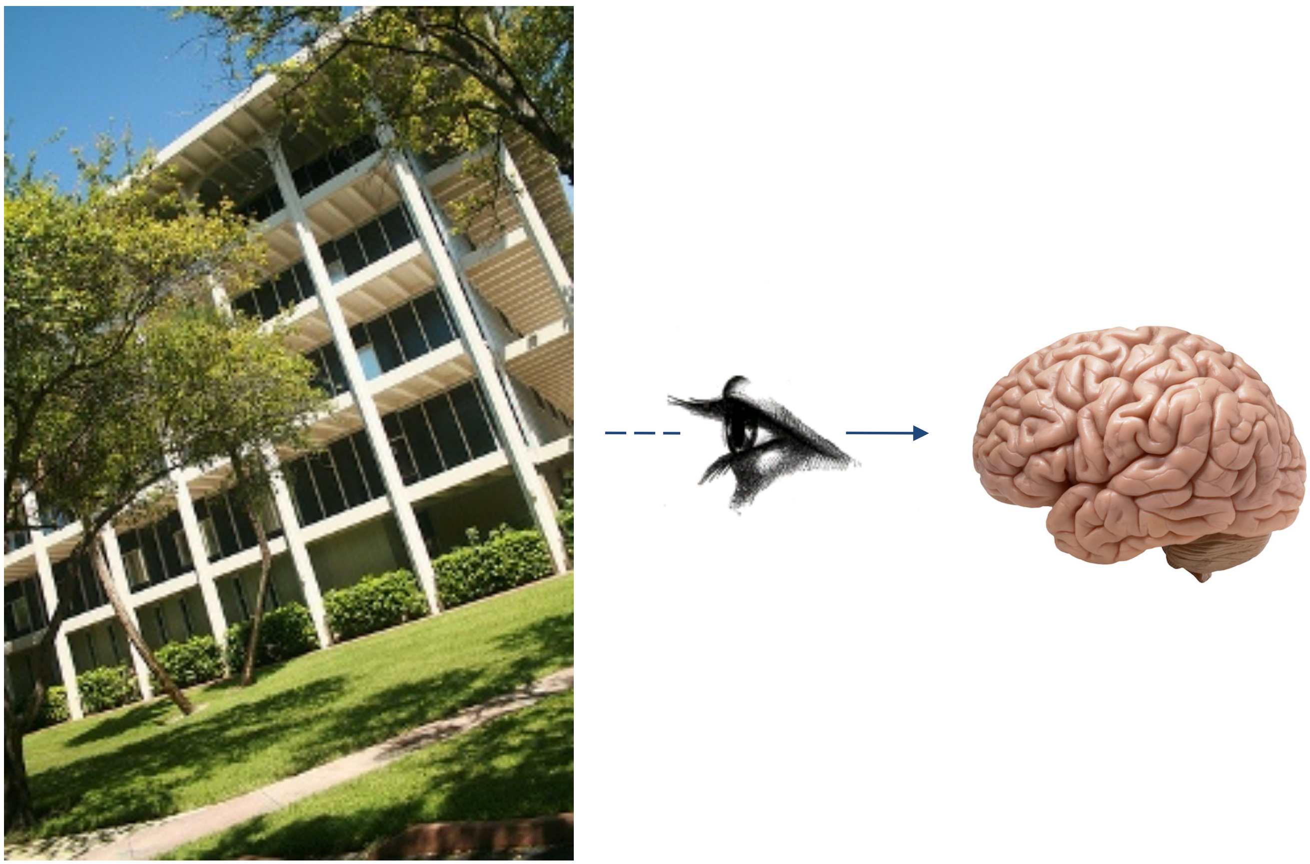
CSC317 House Keeping

- Introductions…

  Your major and what do you hope to get out of this course?

In my field… Computational neuroscience

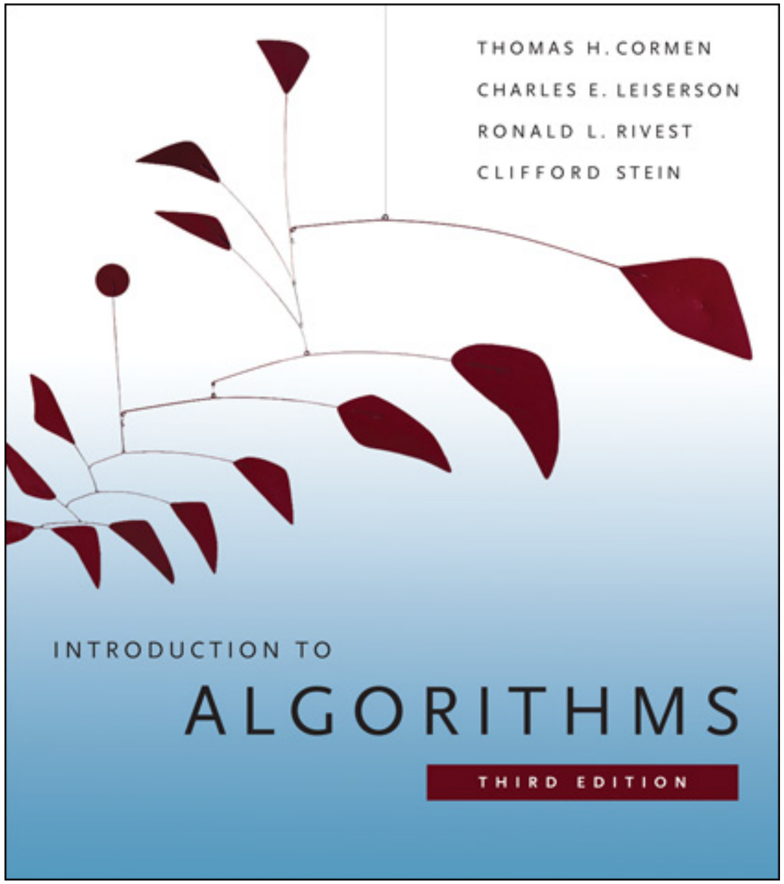
Brain receives input, processes information, and computes outputs. What algorithms does the brain use??



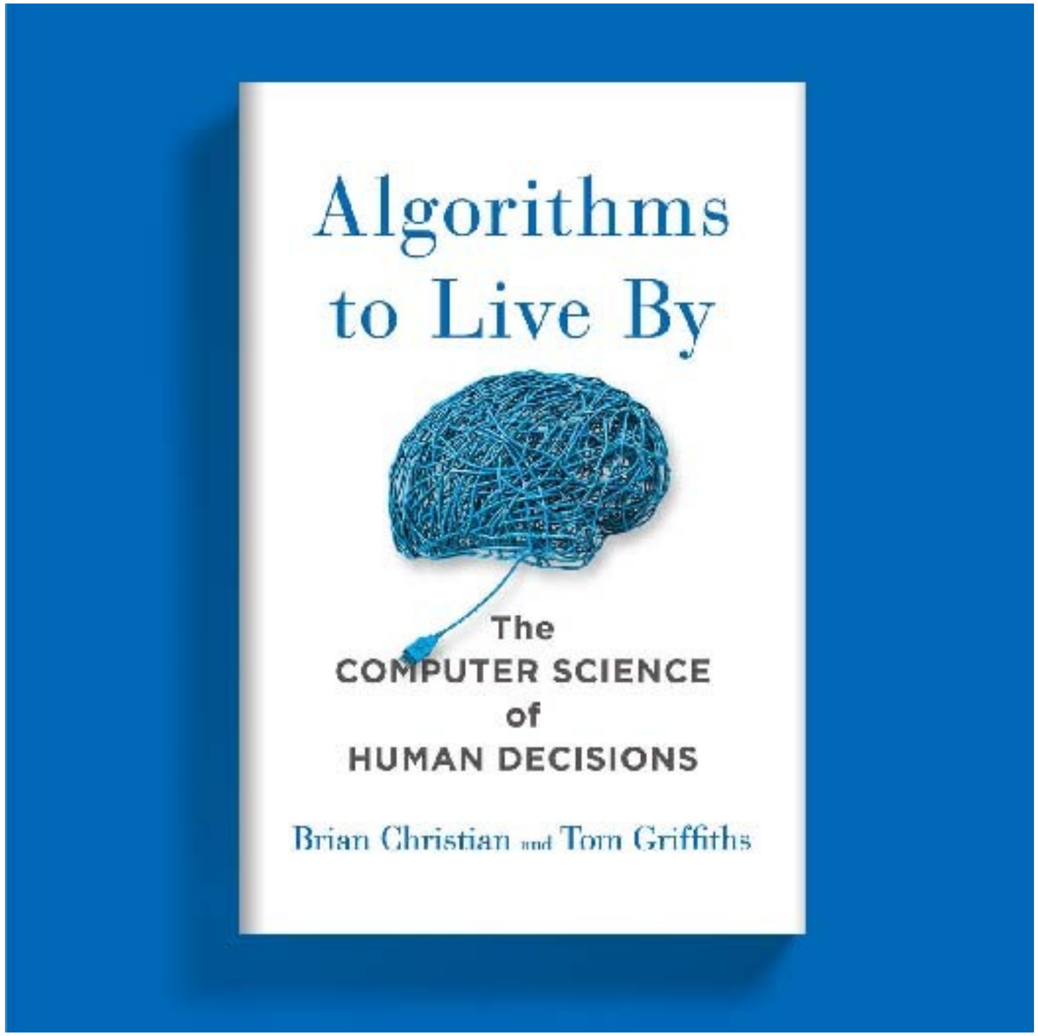
CSC317 House Keeping

- **Course homepage: I will post slides:** http://www.cs.miami.edu/home/odelia/teaching/csc317_fall19/index.html
- My typed slides will be posted on a regular basis; in class develop on the board…
- Odelia Schwartz (odelia at cs miami dot edu). Encouraged to email to make appointment or stop by
- Assignments continue to be on BB
- Continued structure of quizzes (and no final!)

Data Structures and Algorithm Analysis (CSC317)



Optional reading, beyond scope



Algorithmic approaches so far?

Algorithmic approaches so far?

- Divide and Conquer

Algorithmic approaches so far?

- Divide and Conquer

Next:

- Dynamic Programming
- Greedy algorithms

Algorithmic approaches so far?

- Divide and Conquer

Next:

- Dynamic Programming
- Greedy algorithms

Dynamic Programming

- General, powerful
- Problems that may naively have exponential running time, but can be made poynomial (fast!)
- “Programming”?

Dynamic Programming

- General, powerful
- Problems that may naively have exponential running time, but can be made poynomial (fast!)
- "Programming"? Not programming languages; Bellman was interested in planning and decision making. See:

http://www.cs.miami.edu/home/odelia/teaching/csc317_fall19/syllabus/dy_birth.pdf

Dynamic Programming

- General, powerful
- Problems that may naively have exponential running time, but can be made poynomial (fast!)
- "Programming"? Not programming languages; Bellman was interested in planning and decision making. See:

http://www.cs.miami.edu/home/odelia/teaching/csc317_fall19/syllabus/dy_birth.pdf

- Can be thought of as "tabular programming" as in "table." Main approach: hold answers to previous problems already solved in a table, so that can be used again without recomputing.

Dynamic Programming Class Outline

- Examples of applications (motivation)
- Simple example to gain intuition
- Back to applications and more examples (next classes)

Dynamic Programming Class Outline

- Examples of applications (motivation)
- Simple example to gain intuition
- Back to applications and more examples (next classes)

Examples of applications (motivation)

Examples of applications (motivation)

- Computational Biology (genome similarity; also spike similarity; file text similarity)
- Cypher to Thomas Jefferson (will mention)
- Finding shortest path (later)

Examples of applications (motivation)

- Computational Biology (genome similarity)

Strings from alphabet {A, C, G, T}

Example: ACGGAT
CCGCTT

How to determine similarity?

And why? Understanding one genome sequence and its similarity to another can teach us about function…

Examples of applications (motivation)

- Computational Biology (genome similarity)

Strings from alphabet {A, C, G, T}

Example: ACGGAT
CCGCTT

How to determine similarity?

Number of changes from one to another small
Allowed to change character
Find the Longest Common Subsequence

Examples of applications (motivation)

- Computational Biology (genome similarity)

Strings from alphabet {A, C, G, T}

Example: ACGGAT
CCGCTT

What is the Longest Common Subsequence?

Examples of applications (motivation)

- Computational Biology (genome similarity)

Strings from alphabet {A, C, G, T}

Example: ACGGAT
CCGCTT

What is the Longest Common Subsequence?

Answer: 3 CGT

Is answer unique?

Examples of applications (motivation)

- Computational Biology (genome similarity)

Strings from alphabet {A, C, G, T}

Example: ACGGAT
CCGCTT

What is the Longest Common Subsequence?

We easily eye balled answer for these short sequences.
Longer sequences of 500 or more characters?
Brute force solution?

Examples of applications (motivation)

- Computational Biology (genome similarity)

What is the Longest Common Subsequence?
A C C C G G T C G A G T G …
G T C G T T C G G A A T T …

Brute force: Try all subsequences in 1st string and compare to second string…
n=500 then 2^500 possibilities

Pick first character or do not…
Pick 2nd character or do not…
Pick 3rd or do not…
2 * 2 * 2 * 2 …. * 2 (n times)

Examples of applications (motivation)

- Computational Biology (genome similarity)

What is the Longest Common Subsequence?
A C C C G G T C G A G T G …
G T C G T T C G G A A T T …

Brute force:
n=500 then 2^500 possibilities

Pick 1st character or do not…
Pick 2nd character or do not…
Pick 3rd or do not…
2 * 2 * 2 * 2 …. * 2 (n times) = 2^n
Actually need to multiply by length of 2nd string

Examples of applications (motivation)

- Computational Biology (genome similarity)

What is the Longest Common Subsequence?
A C G G A T
C C G C T T

We learned Divide and Conquer. Will this approach work?

Examples of applications (motivation)

- Computational Biology (genome similarity)

  What is the Longest Common Subsequence?
  A C G G A T
  C C G C T T

  We learned Divide and Conquer. Will this approach work?

  Answer: No. Not in a simple way.

  It could for this example, but not generally…
  A C G   G A T
  C C G   C T T
  C G   T

Examples of applications (motivation)

- Computational Biology (genome similarity)

  What is the Longest Common Subsequence?
  C G T G A C
  C G G T T T

  We learned Divide and Conquer. Will this approach work?

  Answer: No. Not in a simple way. Does not find C G T,
  Unless you look across the midline…
  Doesn't work easily here…
  C G T   G A C
  C G G   T T T
  C G

Examples of applications (motivation)

- Computational Biology (genome similarity)

  What is the Longest Common Subsequence?
  C G T G A C
  C G G T T T

  We will learn a Dynamic Programming approach…

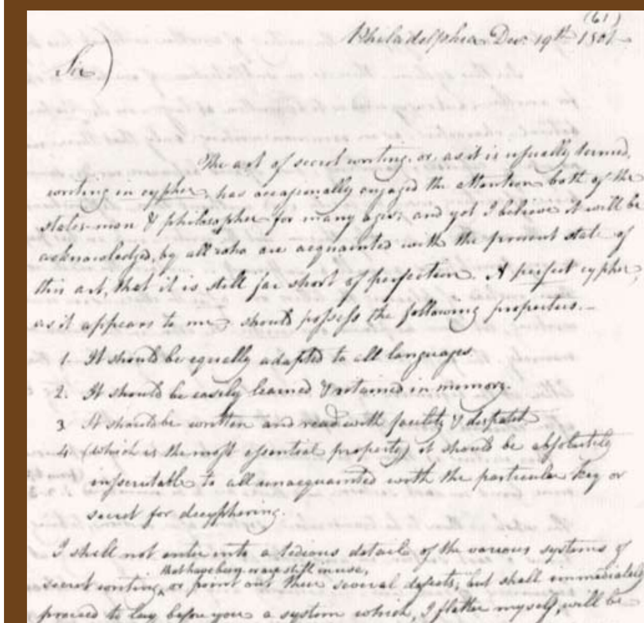
Examples of applications (motivation)

- Spike Similarity…

Examples of applications (motivation)

- Cypher to Thomas Jefferson

http://www.cs.miami.edu/home/odelia/teaching/csc317_fall19/syllabus/cipherJefferson-amsci2009-03S.pdf



(61)

Philadelphia Dec. 19th 1801

Sir

The art of secret writing, or as it is usually termed, writing in cypher, has occasionally engaged the attention both of the states-man & philosopher for many ages; and yet I believe it will be acknowledged, by all who are acquainted with the present state of this art, that it is still far short of perfection. A perfect cypher, as it appears to me, should possess the following properties.

1. It should be equally adapted to all languages.
2. It should be easily learned & retained in memory.
3. It should be written and read with facility & dispatch.
4. (which is the most essential property) it should be absolutely inscrutable to all unacquainted with the particular key or secret for decyphering.

I shall not enter into a tedious detail of the various systems of secret writing that have been or are still in use, or point out their several defects; but shall immediately proceed to lay before you a system which, I flatter myself, will be

Figure 1. On December 19, 1801, Robert Patterson *(far left)*—a professor of mathematics at the University of Pennsylvania—wrote a letter to Thomas Jefferson *(immediate left)* about cryptography. In this letter *(above)*, Patterson described his vision of a "perfect cipher," which required four elements: adaptable to all languages, easy to memorize, simple to perform and inscrutable without the key. Patterson also described an encryption technique that he believed met these criteria. In addition, he included encrypted text, which he said could never be decrypted. There is no evidence that Jefferson was able to decode the text. The author took on Patterson's challenge using techniques that could have been applied—if laboriously—in the early 19th century. (All letter reproductions courtesy of the Library of Congress.)

Examples of applications (motivation)

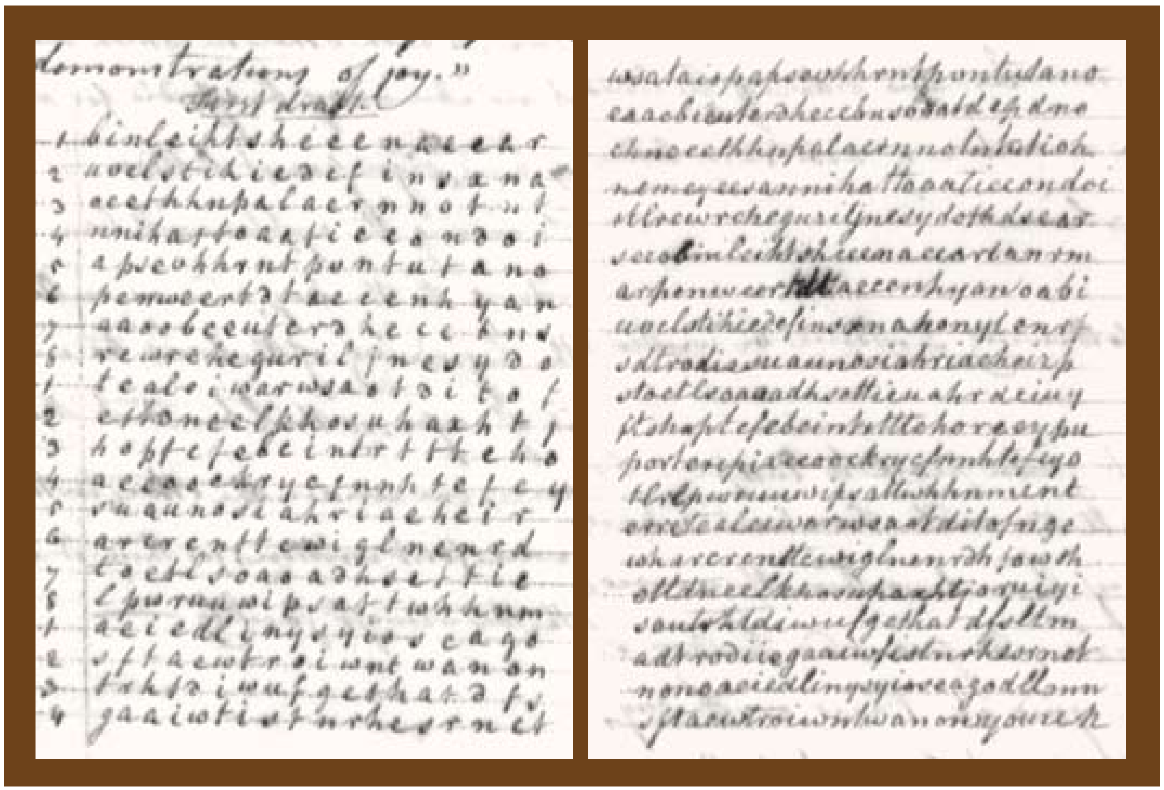
- Cypher to Thomas Jefferson

http://www.cs.miami.edu/home/odelia/teaching/csc317_fall19/syllabus/cipherJefferson-amsci2009-03S.pdf

Examples of applications (motivation)

- Cypher to Thomas Jefferson

http://www.cs.miami.edu/home/odelia/teaching/csc317_fall19/syllabus/cipherJefferson-amsci2009-03S.pdf



Examples of applications (motivation)

- Cypher to Thomas Jefferson

http://www.cs.miami.edu/home/odelia/teaching/csc317_fall19/syllabus/cipherJefferson-amsci2009-03S.pdf

1	binlei	58	wsataispapsevh …
2	uvclst	71	eaao~~o~~bc …
3	oeethh	33	chnoeeth …
4	nnihat	49	nemeyeesannihat …
5	apsevh	83	stlrcwreh …
6	penwee	14	seesbinlei …
7	aaoobc	62	arpenwee …
8	rcwreh	20	uvclst …

Simple example (to build intuition)

- Fibonacci!

Simple example (to build intuition)

- Fibonacci!

Fib(n)

1. If n==0 return 1
2. If n==1 return 1
3. else return Fib(n-1) + Fib(n-2)

Good algorithm??

Simple example (to build intuition)

- Fibonacci!

Fib(n)

1. *If n==0 return 1*
2. If n==1 return 1
3. else return Fib(n-1) + Fib(n-2)

Good algorithm?? Does the job but … no, very wasteful! Why?

Simple example (to build intuition)

Fib(n)

1. *If n==0 return 1*
2. If n==1 return 1
3. else return Fib(n-1) + Fib(n-2)

Good algorithm?? Does the job but … no, very wasteful! Why?

A lot of recomputing …

Simple example (to build intuition)

Fib(n)

1. *If n==0 return 1*
2. If n==1 return 1
3. else return Fib(n-1) + Fib(n-2)

Good algorithm?? Does the job but … no, very wasteful! Why?

Keep repeating computations …
Fib(25) = Fib(24) + Fib(23) …
Fib(24) = Fib(23) + Fib(22)…

Simple example (to build intuition)

Fib(n)

1. *If n==0 return 1*
2. If n==1 return 1
3. else return Fib(n-1) + Fib(n-2)

Recursion tree on the board…

Simple example (to build intuition)

Fib(n)

1. *If n==0 return 1*
2. If n==1 return 1
3. else return Fib(n-1) + Fib(n-2)

See animation:

https://www.cs.usfca.edu/~galles/visualization/DPFib.html

Simple example (Fibonacci)

Summary so far:

- Overlapping subproblems (lots)!
- Solution to big problem can be constructed from solutions to subproblems
- Example of type of problems that can be solved with Dynamic Programming

Simple example (Fibonacci)

Dynamic Programming Fibonacci:

- Main idea: Save in dictionary (e.g., array) subproblems already solved, so no need to recompute

- Memoization: from memo pad or memory …
  funky name…

Fibonacci Memoized Dynamic Programming

On the board…

a. Initialization: Let mem be a new array with values initialized to minus infinity

Fibonacci Memoized Dynamic Programming

a. Initialization: Let mem be a new array with values initialized to minus infinity

b. Fib(n) //Memoized Dynamic Programming

```
1. If mem[n]>=0
2.     return mem[n]    //if already previously computed in memo pad
3. if n==0 return 1
4. if n==1 return 1
5. else f = Fib(n-1) + Fib(n-2) //otherwise compute and save value
6.     mem[n] = f  //save value in memo pad
7.     return f
```

Fibonacci Memoized Dynamic Programming

Plot tree: On the board…

- Run time proportional to n
- Second time encountered, just use memoized result…
- Cuts off whole recursion subtrees!

# of subproblems: n (size of array)
work per subproblem: constant

Fibonacci Memoized Dynamic Programming

Plot tree: On the board…

- Run time proportional to n
- Second time encountered, just use memoized result…
- Cuts off whole recursion subtrees!

See online by Galles:
https://www.cs.usfca.edu/~galles/visualization/DPFib.html

See online by Rosenberg:
http://www.cs.miami.edu/home/odelia/teaching/fib2019.html

Summary:
Recursion + memoization

Another Fibonacci Dynamic Programming (bottom-up)

Fib(n) //Bottom-up Dynamic Programming
1. Let mem[0..n] be a new array
2. mem[0] = 1
3. mem[1] = 1
4. For i=2 to n
5.   mem[i] = mem[i-1] + mem[i-2]
6. return mem[n]

Another Fibonacci Dynamic Programming (bottom-up)

Question: Is bottom-up algorithm the same or different from the previous recursive memoized solution?

See online by Galles:

https://www.cs.usfca.edu/~galles/visualization/DPFib.html

Another Fibonacci Dynamic Programming (bottom-up)

Question: Is bottom-up algorithm the same or different from the previous recursive memoized solution?

See online by Galles:
https://www.cs.usfca.edu/~galles/visualization/DPFib.html

Answer: One for loop, complexity proportional to n
Equivalent solution to recursive memoization (same things
Happen in same order; but in bottom-up we know and
set the order in advance)

Another Fibonacci Dynamic Programming (bottom-up)

Question: If this is how we were first taught Fibbonacci, why bother with naïve inefficient recursion, memoized solution, etc. first?

Answer: Other problems initially less intuitive, but approach will be similar (think back to Genome question)

Dynamic Programming so far

1. Overlapping subproblems (same subproblems solved over and over again
2. Solution to big problem constructed from solutions to smaller subproblems (optimal substructure; more on later)
3. To make algorithm more efficient, we either **memoized (saved solutions to smaller subproblems in a table) as we recursed;** or we saved solutions to subproblems **bottom-up.** These turned out equivalent.

Dynamic Programming so far

Question: Both Dynamic Programming and Divide & Conquer have recursive solutions. But they are different. Why?

Dynamic Programming so far

Question: Both Dynamic Programming and Divide & Conquer have recursive solutions. But they are different. Why?

Answer: For instance, Divide & Conquer doesn't have overlapping subproblems…

Next

- In Fib clear what the smaller subproblems are, and how knowing their solution solves the bigger problem
- Start to build intuition with more complex problems, starting from genome similarity and Longest Common Subsequence…