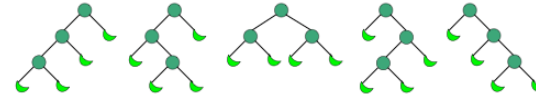


## Algorithms and Idioms

- How do you avoid calculating the same thing over and over and over and over and ...
  - Does it matter?
  - Recalculate v Retrieve: tradeoff of ?
  - Cache/Caching Memoize/Memoization
- Numbers represent state, extend one step at a time
  - How many ways can we get to a corner in a grid?
    - What subproblem helps answer the question
  - Sometimes related to memoizing, sometimes to dynamic programming

## Catalan number

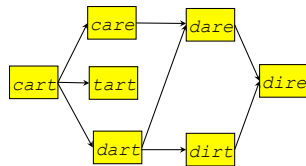
- How many trees are there with N nodes?
  - We can enumerate for a fixed N, e.g., 3 or 4 or ...
  - If we know the number for 1,2,...,N-1, answer for N?



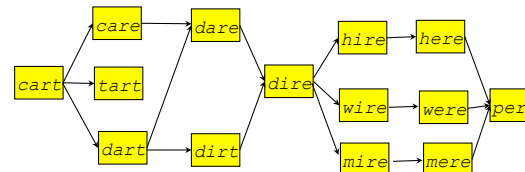
- Try recursive solution dictated by observation
  - Why is it slow? How can we make it fast?
  - Recursion meets memoization!

## Digression: word ladders

- How many ladders from *cart* to *dire* as shown?
  - Enqueue *dare* more than once?
  - Downside? Alternative?
- We want to know number of ladders that end at *W*.
  - What do we know initially?
  - When we put something on the queue, what do we know?
  - How do we keep track?
- Similar to tree-counting?



## Word Ladder: more details



- # ladders that end at *dare*
  - At each word *W*
- Ladder length to *W*
  - Calculable from??
- Two maps

- Dequeue *s*
  - foreach *W* one-away
  - if not-seen ???
  - else ???

## Alan Kay

- Turing award 2003
  - OO programming, Dynabook
- “The best way to predict the future is to invent it”
- “American’s have no past and no future, they live in an extended present.”



I think the main thing about doing ...any kind of programming work, is that there has to be some exquisite blend between beauty and practicality. There's no reason to sacrifice either one of those, and people who are willing to sacrifice either one of those, I don't think really get what computing is all about.

11.5

## Digression: Bogglescore APT

- How many ways to find Aha?!
  - Each one scores points
  - Related to genomics problem
- Number of ways that AHA ends at (2,1)?
  - What do we know initially?
  - When we extend search what do we know?
  - How do we keep track?
- Why do we need to avoid revisits?
  - Caching, memoizing needed!

A	H	A	A
A	A	H	A
H	H	A	A
A	A	H	A

CPS 100, Fall 2009

11.6

## Digression: Bogglescore APT

- # times we can find: AAAAA...AAA
  - # 50-step paths in grid, why? Big? Small?
  - Calculate # 49 step paths at each spot
    - Foreach cell, sum neighbors
    - If recursive, what happens?
- Memoize on row, column, word-length
  - For each word, clear cache/memo
  - For every row, column
    - If [r][c] == first char, recurse
- String shorter by 1
  - Toward length 1, stop and return 1
  - If seen [r][c][length] return value
  - When do we return zero?

		A	A	A	A
		A	A	A	A
		A	A	A	A
		A	A	A	A
-1	-1	-1	-1		
-1	-1	-1	-1	-1	
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	
		-1	-1	-1	-1

CPS 100, Fall 2009

11.7

## Bogglescore alternative

- Last solution was lexicon-first with memoizing
  - Visit every cell, but don't revisit with identical substring
- What about board-first solution?
  - When we visit a cell, we're working with word-so-far
  - Avoid revisiting cell with same word-so-far
    - Memoize on word/row/col, make this Map-able
    - Hash on word, row, and col
    - Compare based on word, row, and col
  - Don't recurse if not a prefix
  - Store score if word-so-far is a word, still recurse

CPS 100, Fall 2009

11.8