15-112
Lecture 2

Recursion

Instructor: Pat Virtue

Tuesday Logistics

As you walk in

Quiz will start at the beginning of lecture

- Have pencil/pen ready
- Silence phones

Quiz

Before we start

- Don’t open until we start
- Make sure your name and Andrew ID are on the front
- Read instruction page
- No questions (unless clarification on English)

Additional info

- 25 min

Fractals

Mandelbrot set

https://www.youtube.com/watch?v=u1pwtSBTnPU

Announcements

Democracy Day

- Tue 11/7 – No class

TP

- Ideation meetings
- Special topic session
- Scaffolded project – Bee project

HW9

- VS Code graphics exercise → Turn in on Canvas

Thursday Logistics

Announcements

Quiz 8

Poll 1

What is the big O of the following function, which takes a list of length N? Note that some parts of the code are intentionally blanked out.

A. O(1)
B. O(2**N)
C. O(N**0.5)
D. O(N*log(N))
E. O(N**2)
F. O(N**3)
G. Need more information to be sure

```
def f(L):
    result = []
    for i in range(_______):
        for j in range(_______):
            k = L[i] + L[j] result.append(k)
    return result
```

Poll 2

Which of the following may require Python to visit all N elements in the list data, assuming N = len(data)? Select ALL that apply.

A.
```
for x in data:
    print(x)
```
B.
```
for i in range(len(data)):
    x = data[i]
    print(x)
```
C.
```
if x in data:
    print(“Found it”)
```
D. `x = data[-1]`

E. `x = max(data)`

F. None of the above

Poll 3

Assume print(s) prints `{2, 4, 6, 8, ???}` for some set `s`.

Which of the follow will are possible replacements for `???` for some set?

Select ALL that apply.

A. 1

B. [1]

C. 2

D. 'two'

E. 10

F. {10}

G. Need more information to be sure

Announcements

Next Week

- Tue 11/7: No class (Democracy Day)
- Thu 11/9: Quiz 9
- HW10

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Week 9					Recursion		HW9 due Hack112
Week 10	Hack112		No class Preread10		Quiz 9 OOP Part1		HW10 due
Week 11			No quiz10		Midterm 2		

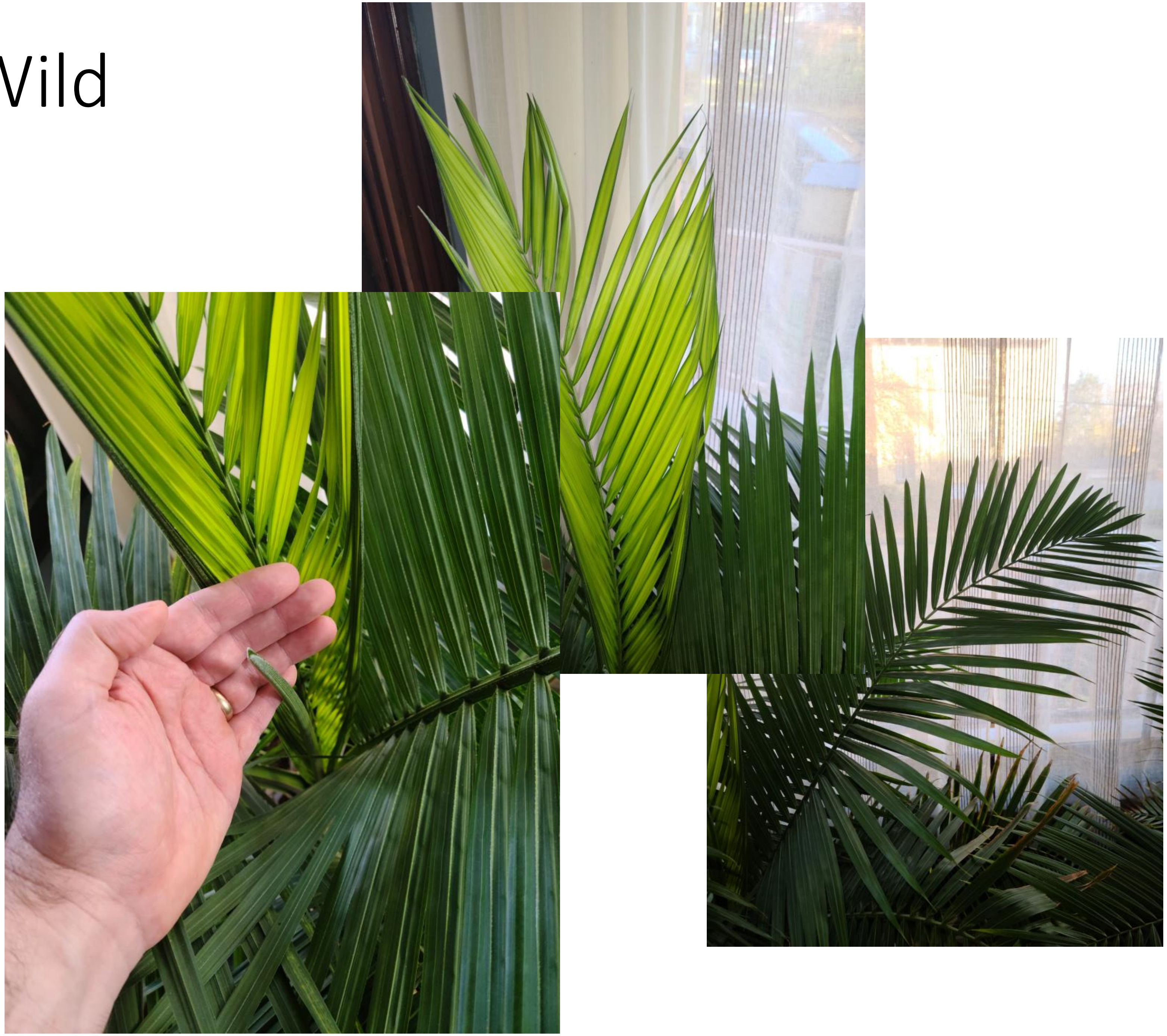
TP

- Special topic sessions will be posted today

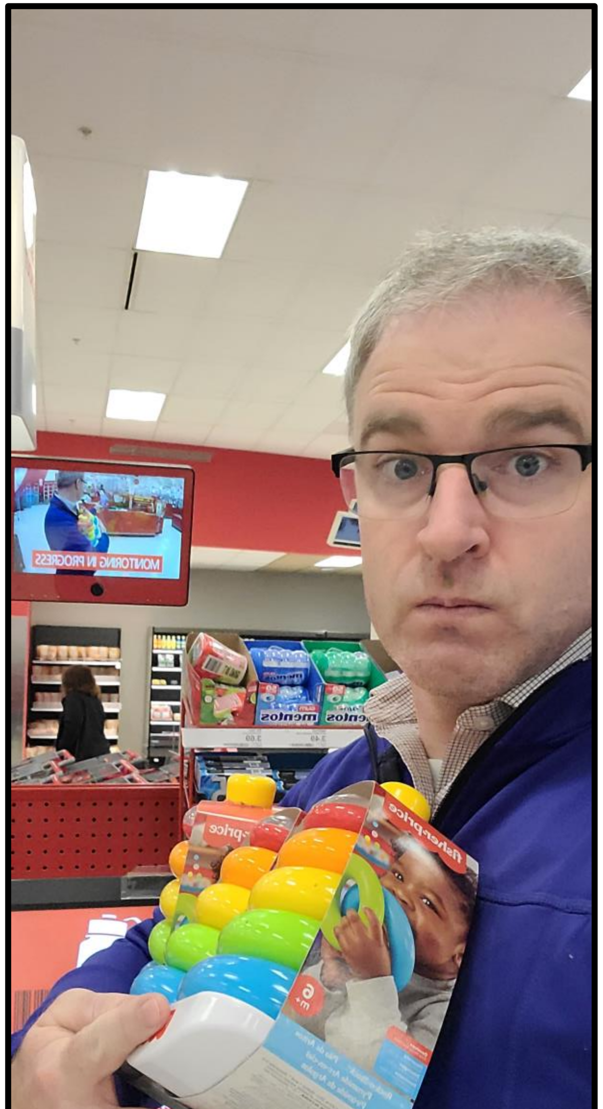
Recursion

Recursion in the Wild

Recursion in the Wild



Recursion in the Wild



Recursion in the Wild


```html
<div class="row col-lg-10 col-lg-offset-1">
  <div id="overview">

    <h1>Overview</h1>

    <div class="well bs-component">
      <form class="form-horizontal">
        <div class="form-group">
          <label class="col-sm-2 control-label">Units</label>
          <div class="col-sm-10">
            <p class="form-control-static">12</p>
          </div>
        </div>
        <div class="form-group">
          <label class="col-sm-2 control-label">Department</label>
          <div class="col-sm-10">
            <p class="form-control-static">
              <a href="http://www.csd.cs.cmu.edu/" target="_blank">Computer Science</a>
            </p>
          </div>
        </div>
        <div class="form-group">
          <label class="col-sm-2 control-label">Prerequisites</label>
          <div class="col-sm-10"><p class="form-control-static">None</p>
          </div>
        </div>
        <div class="form-group">
          <label class="col-sm-2 control-label">Textbook</label>
          <div class="col-sm-10">
            <p class="form-control-static">None. Course notes included on course website.</p>
          </div>
        </div>
        <div class="form-group">
          <label class="col-sm-2 control-label">Description</label>
          <div class="col-sm-10">
            <p class="form-control-static">
              A technical introduction to the fundamentals of programming with an emphasis
              on producing clear, robust, and reasonably efficient code using top-down
              design, informal analysis, and effective testing and debugging. Starting
              from first principles, we will cover a large subset of the Python
              programming language, including its standard libraries and programming
              paradigms.
            </p>
            <p class="form-control-static">
```

General Recursive Form

```
def recursiveFunction():
    if (this is the base case):
        do something non-recursive
    else:
        do something recursive
```

Recursive thinking

Suggestion: start with the recursive case

- How can you reduce the problem into smaller problem(s) that have the same structure as the original?
- Assume (magically) that next recursive cases will work

Recursive thinking (and recursive functions)

Count digits??

```
def countDigits(number):
```

Recursive thinking (and recursive functions)

Word search??

```
def wordSearch(board, word):
    (rows, cols) = (len(board), len(board[0]))
    for row in range(rows):
        for col in range(cols):
            result = wordSearchFromCell(board, word, row, col)
            if (result != None):
                return result
    return None
```

Recursion Examples

- Recursive case
- Base case
- Recursion errors
- Call Stack
- Visualizing recursion
- Debugging recursion

Example: Factorial

Example: Factorial

Some Recursion Issues

Debugging alternatingCase

```
def alternatingCase(s):


    # assume s is at least of length 1:
    if len(s) == 1:

        return s[0].upper()
    else:
        last = s[-1]
        rest = s[:-1]
        if alternatingCase(rest)[-1].isupper():

            return alternatingCase(rest) + last.lower()
        else:

            return alternatingCase(rest) + last.upper()
```

Example: Fibonacci

Towers of Hanoi

Goal: Move stack to a different peg

Restrictions

- One piece at a time
- Can't put bigger piece on top of smaller

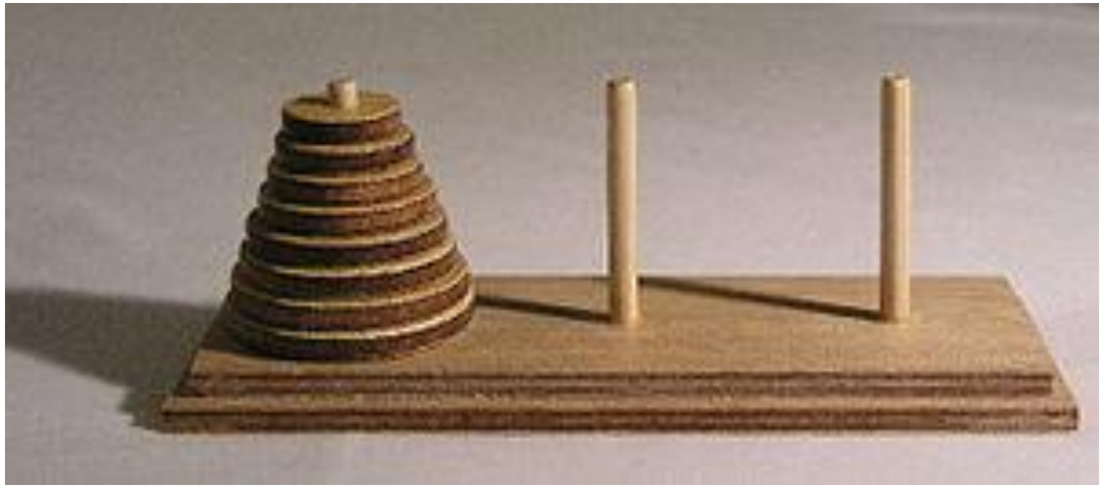


Image (left): https://commons.wikimedia.org/wiki/File:Tower_of_Hanoi.jpeg

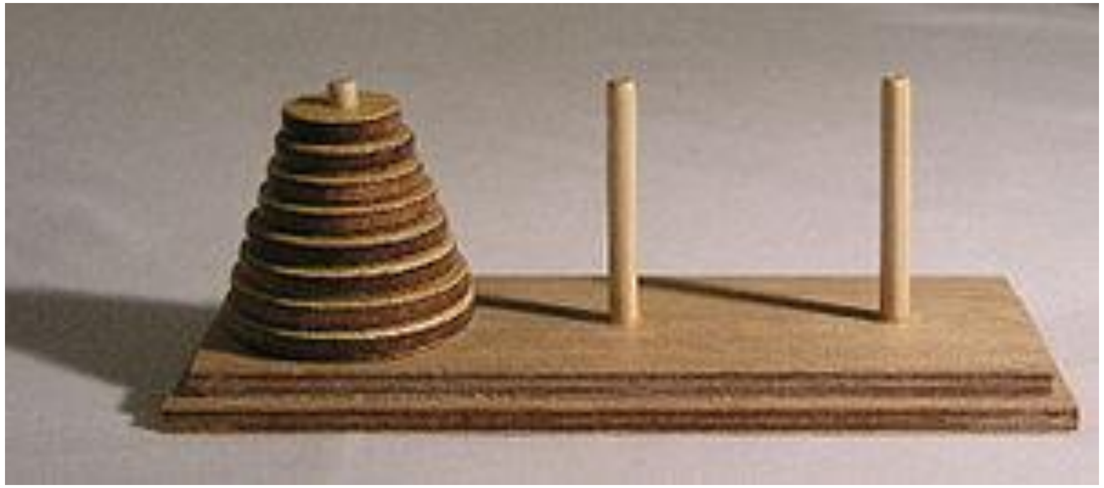
Reminder General Recursive Form

```
def recursiveFunction():
    if (this is the base case):
        do something non-recursive
    else:
        do something recursive
```

Towers of Hanoi

Recursive case

- Let’s start with magic!

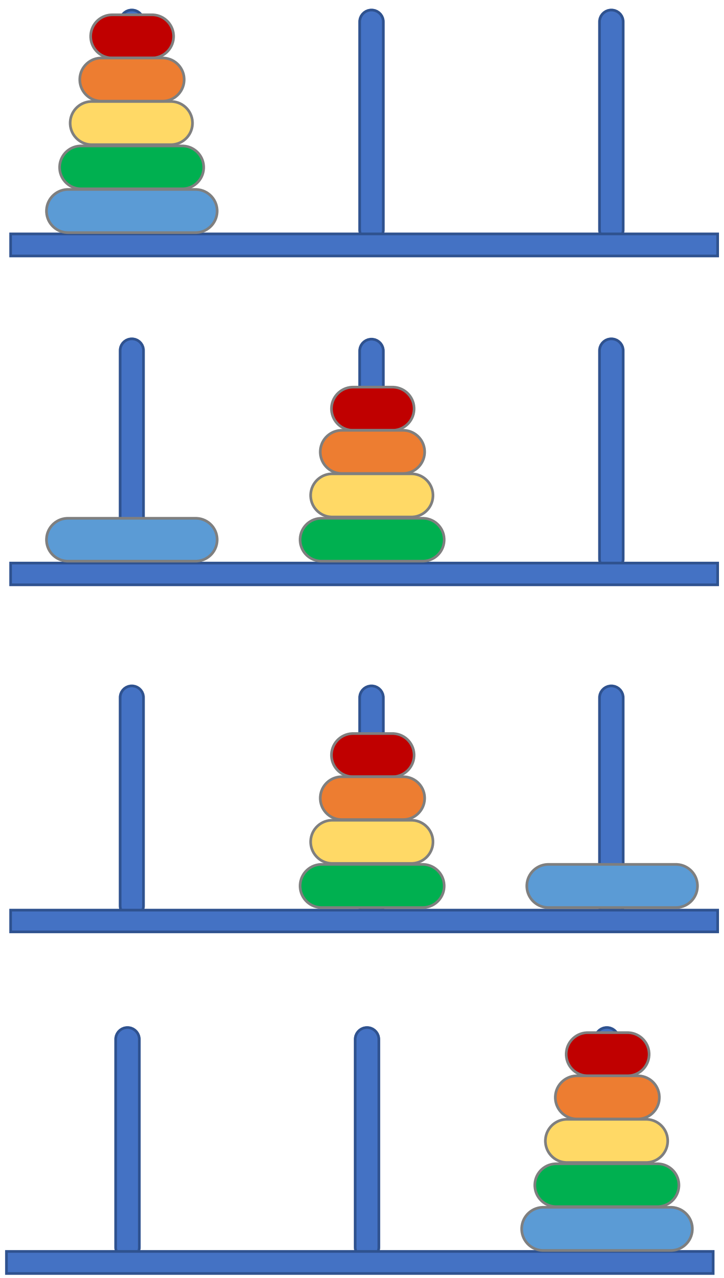


Towers of Hanoi

Recursive case

- Let's start with magic!

```
import magic # For now :)

def move5(start, end, temp):

    # Move 5 pieces from start to end

    magic.move4(start, temp, end)

    print(f"Move piece from {start} to {end}")

    magic.move4(temp, end, start)
```

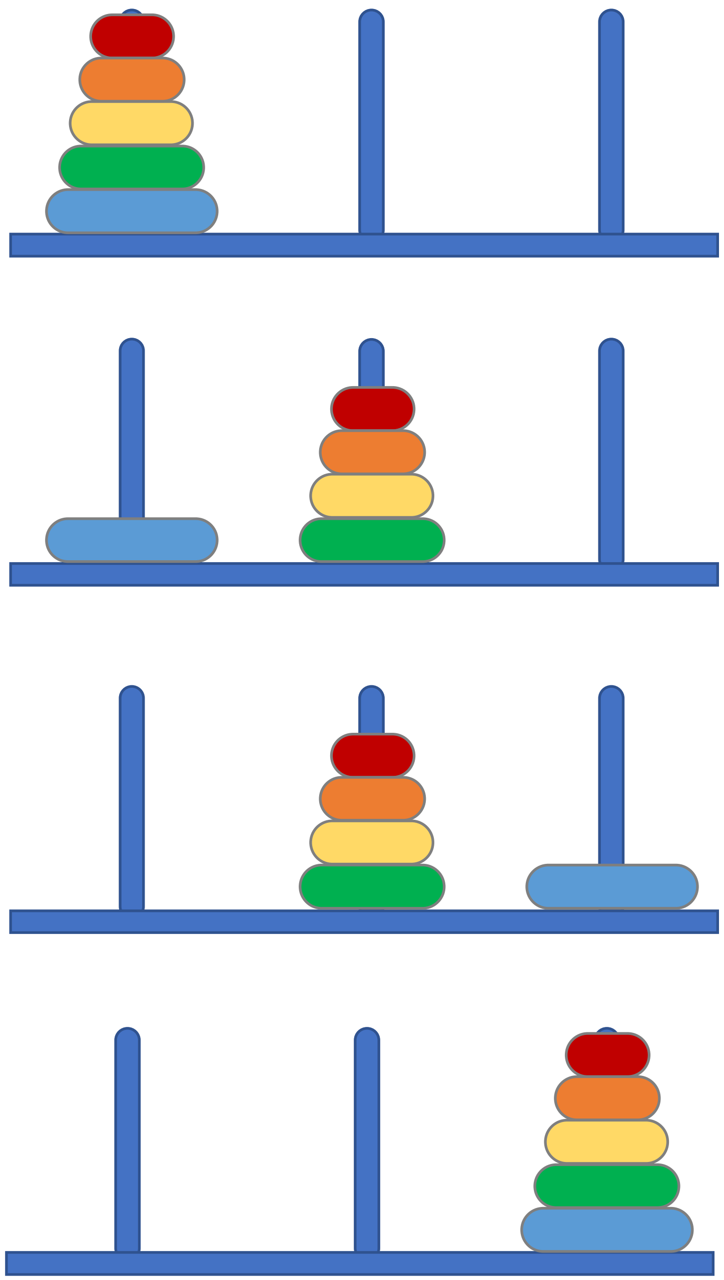


Towers of Hanoi

Recursive case

- ~~Let’s start with magic!~~

```
import magic # For now :)

def move(start, end, temp):
    # TODO Base case
    # Move n pieces from start to end

    move(n-1 start, temp, end)

    print(f"Move piece from {start} to {end}")

    move(n-1, temp, end, start)
```



Revisit Merge Sort

Merge sort: $O(N \log N)$

Merge concept:

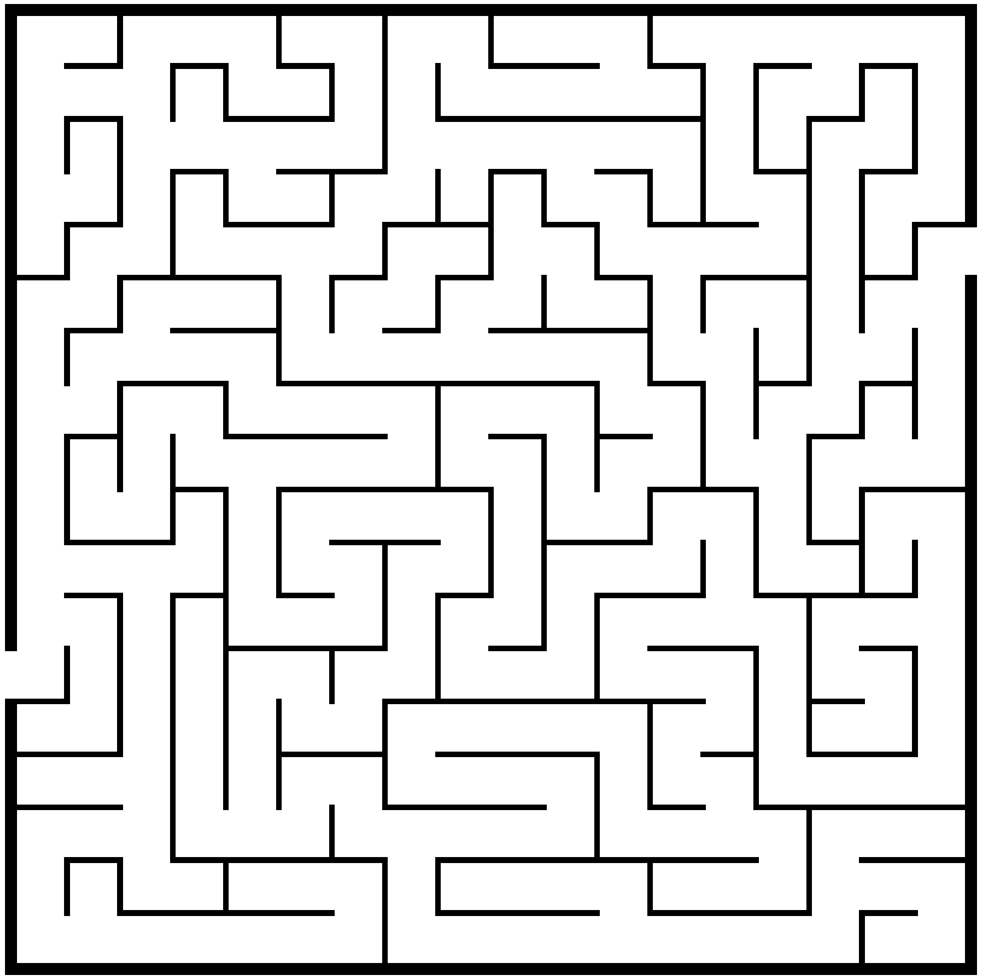
Assume you had two piles that were already independently sorted.

Could you shuffle them together into one sorted pile in O(N)?

```
def mergesort(L):
  if len(L) < 2:
    return L
  else:
    mid = len(L)//2
    left = mergesort(L[:mid])
    right = mergesort(L[mid:])
    return merge(left, right)
print(mergesort([1,5,3,4,2,0]))
```

Backtracking

Incredibly generic problem-solving algorithm



Backtracking: Word chain

List of words

Return an ordered list of words such that

- Last letter of each word is the first letter of the next word

Debug output should matched tree!

```
CHAIN: [], REMAINING: ['goose', 'dog', 'elk', 'toad']
    |CHAIN: ['goose'], REMAINING: ['dog', 'elk', 'toad']
    |    |CHAIN: ['goose', 'elk'], REMAINING: ['dog', 'toad'
    |    |Result: False
    |Result: False
    |CHAIN: ['dog'], REMAINING: ['goose', 'elk', 'toad']
    |    |CHAIN: ['dog', 'goose'], REMAINING: ['elk', 'toad'
    |    |    |CHAIN: ['dog', 'goose', 'elk'], REMAINING: ['t
    |    |    |Result: False
    |    |Result: False
    |Result: False
    |CHAIN: ['elk'], REMAINING: ['goose', 'dog', 'toad']
    |Result: False
```

Backtracking: Word chain

solve(chain, words)

1. If no more words

    Return chain as solution!

2. For each valid action
    a) Apply action
    b) Recurse: result = solve(chain, words)
    c) If result is success

        Return result

    Else

        Undo action

3. Return failure

Backtracking: N-Queens Example

Backtracking: N-Queens Example

N-by-N chessboard

Place exactly N queen pieces on the board, such that no queens are in positions to attack each other

- Queens can move any number of spaces:
  - Horizontally
  - Vertically
  - Diagonally

Backtracking: N-Queens Example

solve(board)

1. If all Qs placed

   Return board as solution!

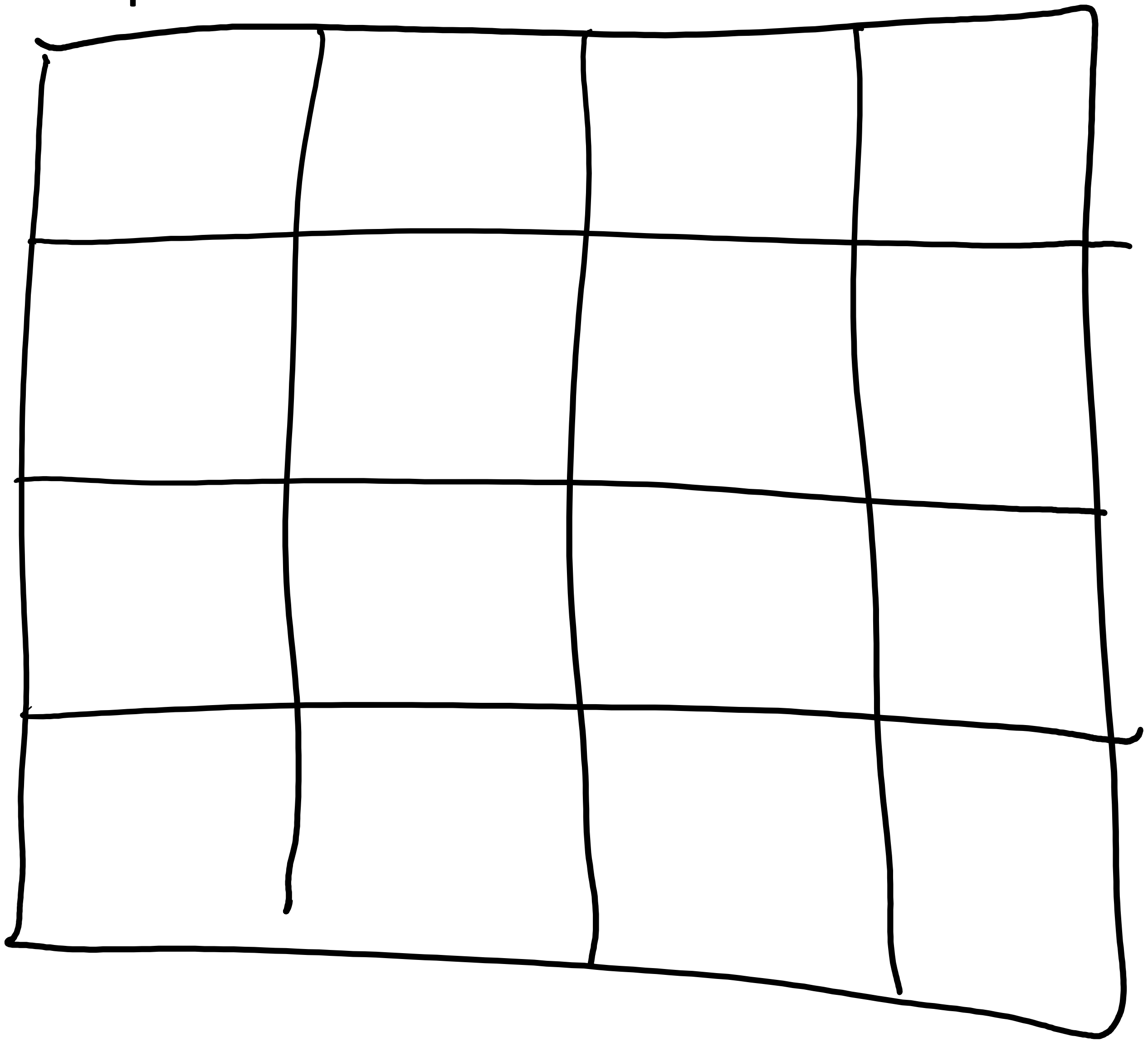
2. For each valid action
   a) Apply action
   b) Recurse: result = solve(board)
   c) If result is success

      Return result

   Else

      Undo action

3. Return failure



Backtracking: N-Queens Example

Backtracking: N-Queens example

Code demo

https://www.cs.cmu.edu/~112/notes/notes-recursion-part2.html#nQueens

Backtracking: Solving maze example

Start: top-left

Goal: bottom-right

Strategy

- Path: Keep ordered list of locations representing the current path
- Visited: Avoid revisiting same locations by storing
- Try actions in order: N, S, E, W
- Recursively solve from next location

Backtracking: Solving maze example

solve(maze, path, visited)

1. If at goal

   Return path as solution!

2. For each valid action
   a) Apply action
   b) Recurse:

      result = solve(maze, path, visited)
   c) If result is success

      Return result

      Else

      Undo action

3. Return failure

Backtracking pattern

solve(maze, path, visited) — Maze

1. If at goal

   Return path as solution!

2. For each valid action
   a) Apply action
   b) Recurse:

      result = solve(maze, path, visited)

   c) If result is success

      Return result

      Else

      Undo action

3. Return failure

solve(board) — N-Queens

1. If all Qs placed

   Return board as solution!

2. For each valid action
   a) Apply action
   b) Recurse:

      result = solve(board)

   a) If result is success

      Return result

      Else

      Undo action

3. Return failure

Backtracking: Performance

Map coloring example

Goal: color all states with {**red**, **green**, **blue**} such that adjacent states have different colors.

Classic example: Australia

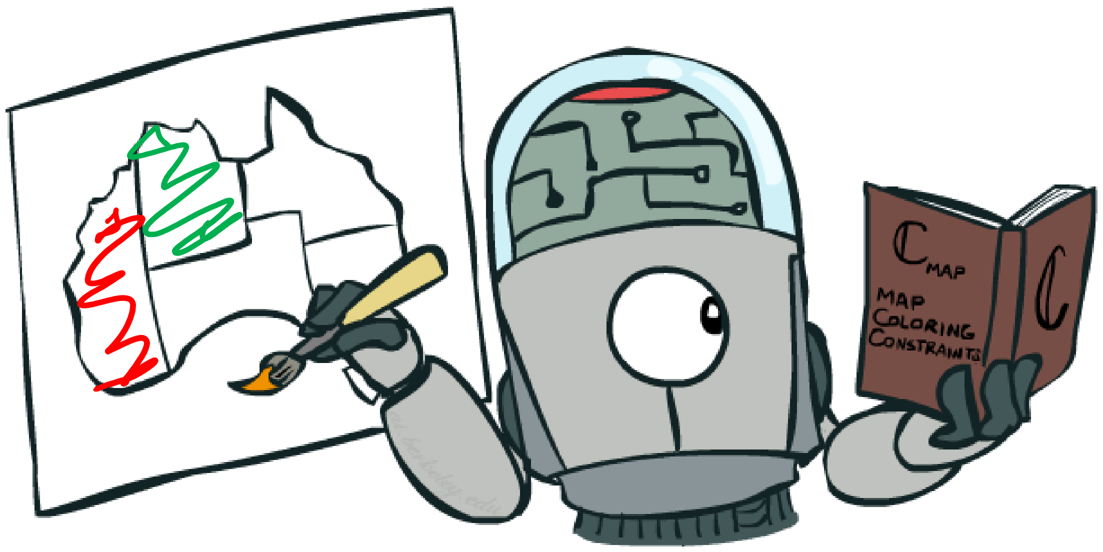


Image credit: ai.berkeley.edu, Ketrina Yim

Backtracking: Performance

Map coloring example

"for each valid action"

But, what are the actions??

Backtracking: Performance

Map coloring example

"for each valid action"

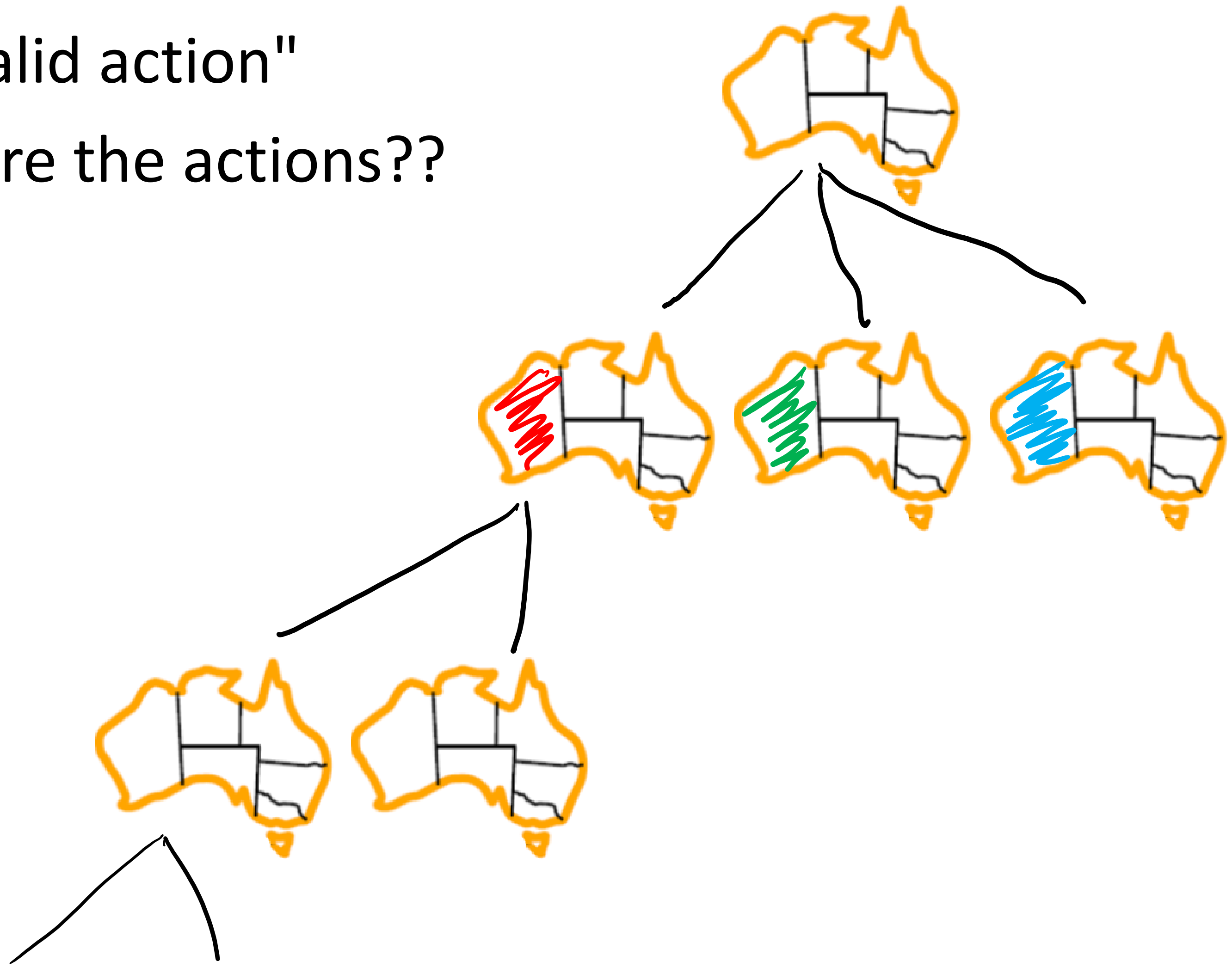
But, what are the actions??

Backtracking: Performance

Map coloring example

"for each valid action"

But, what are the actions??



Fractals!