Recursive Trees

CS 106B

Programming Abstractions
Fall 2016
Stanford University
Computer Science Department

Announcements

- Assignment 2 due Saturday
- Assignment 3 goes out today (due Oct 24th)
- CS for Social Good has a winter class CS 51

Assignment 3

META

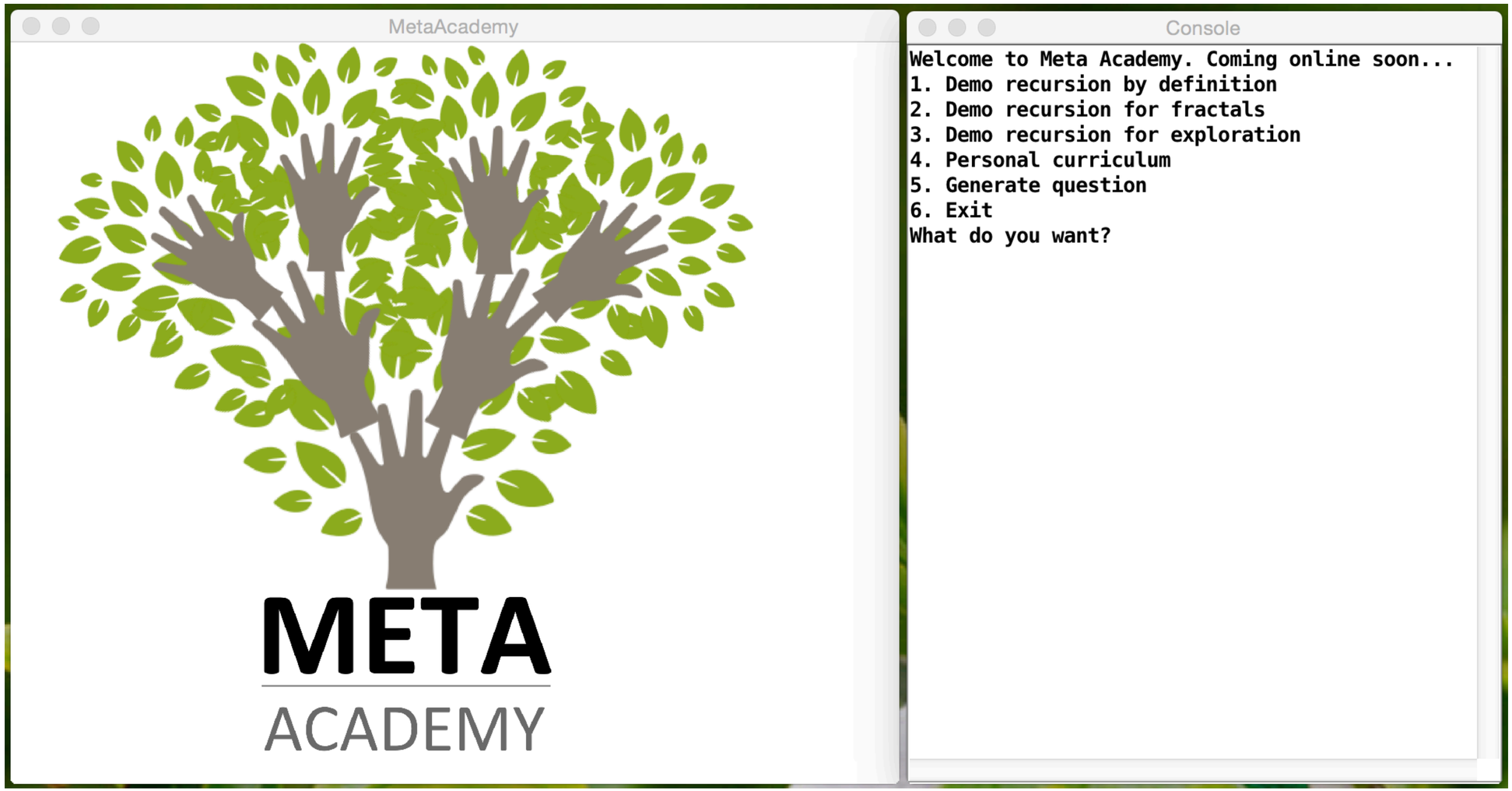
ACADEMY

Due: Oct 24th

YEAH: Monday 7-8pm

Location: TBA

Assignment 3



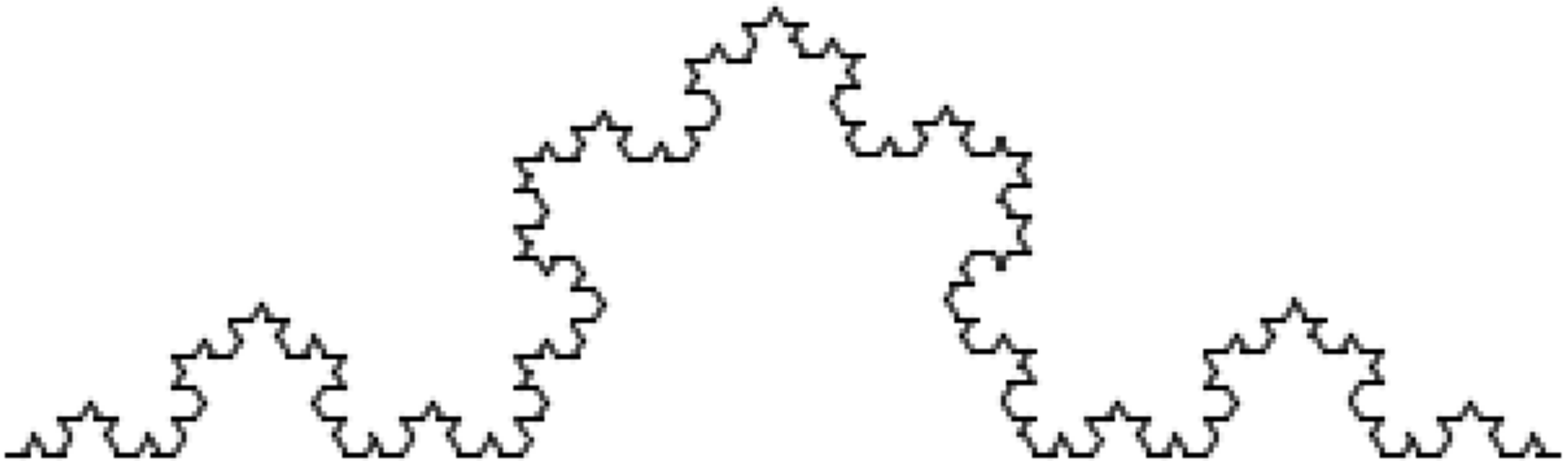
In order to learn recursion, you will teach recursion.
In order to teach recursion, you must first learn recursion.

Recursion by Definition

$$x^0 = 1$$

$$x^n = x \cdot x^{n-1}$$

Fractals



Today's Goal

1. Learn to see trees in recursion

2. Be ready for assn #3

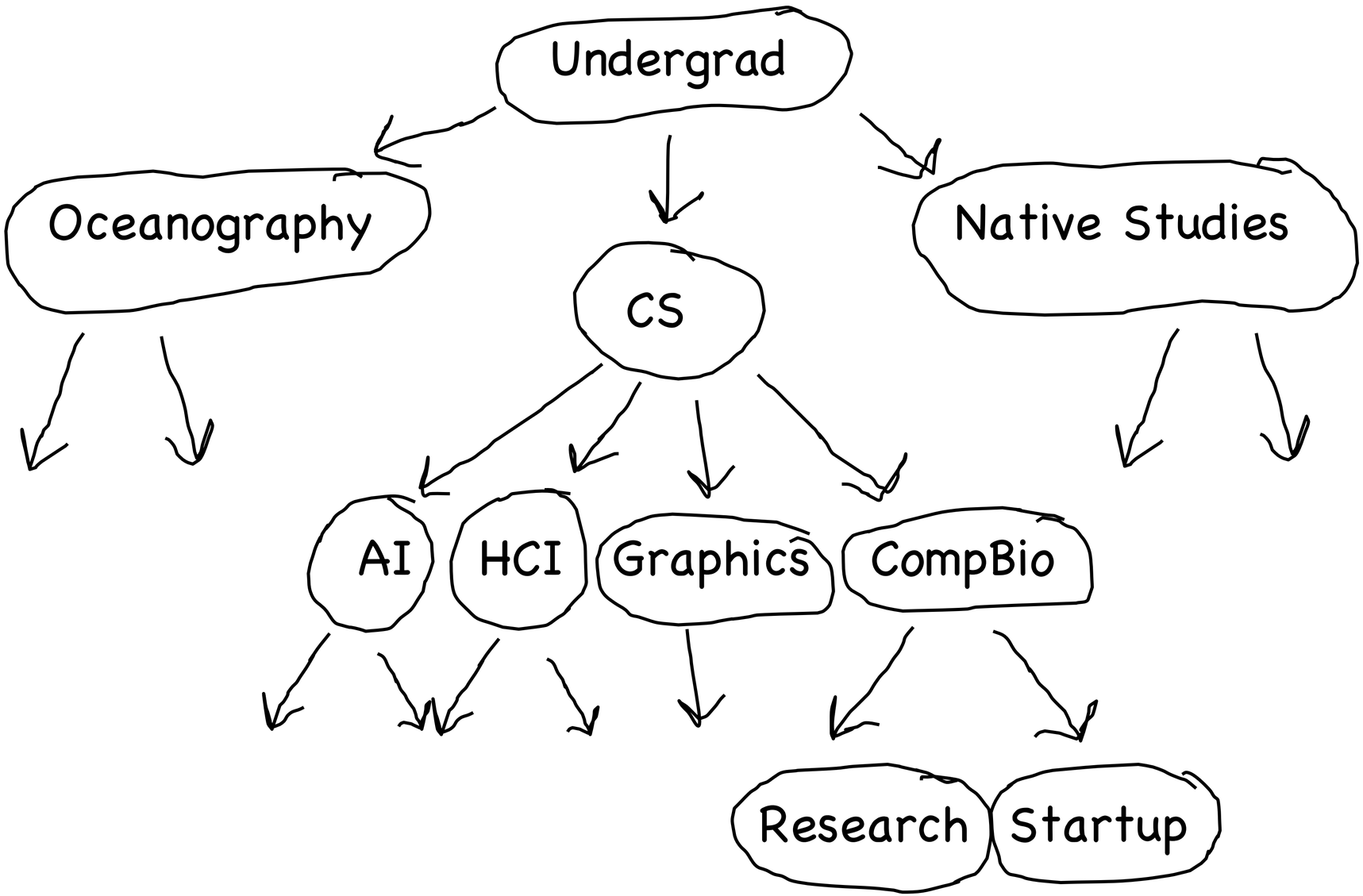
Problems with the Same Underlying Form

1. List all **permutations** of a set of elements
2. Output the path of all files in a **folder**.
3. Find all words that are "**reducible**"
4. Play a game of knight's tour **solitaire**
5. Find the optimal "alignment" of two strands of **DNA**

They have a similar underlying representation
and they can all be solved with recursion

Lets talk about your life decisions

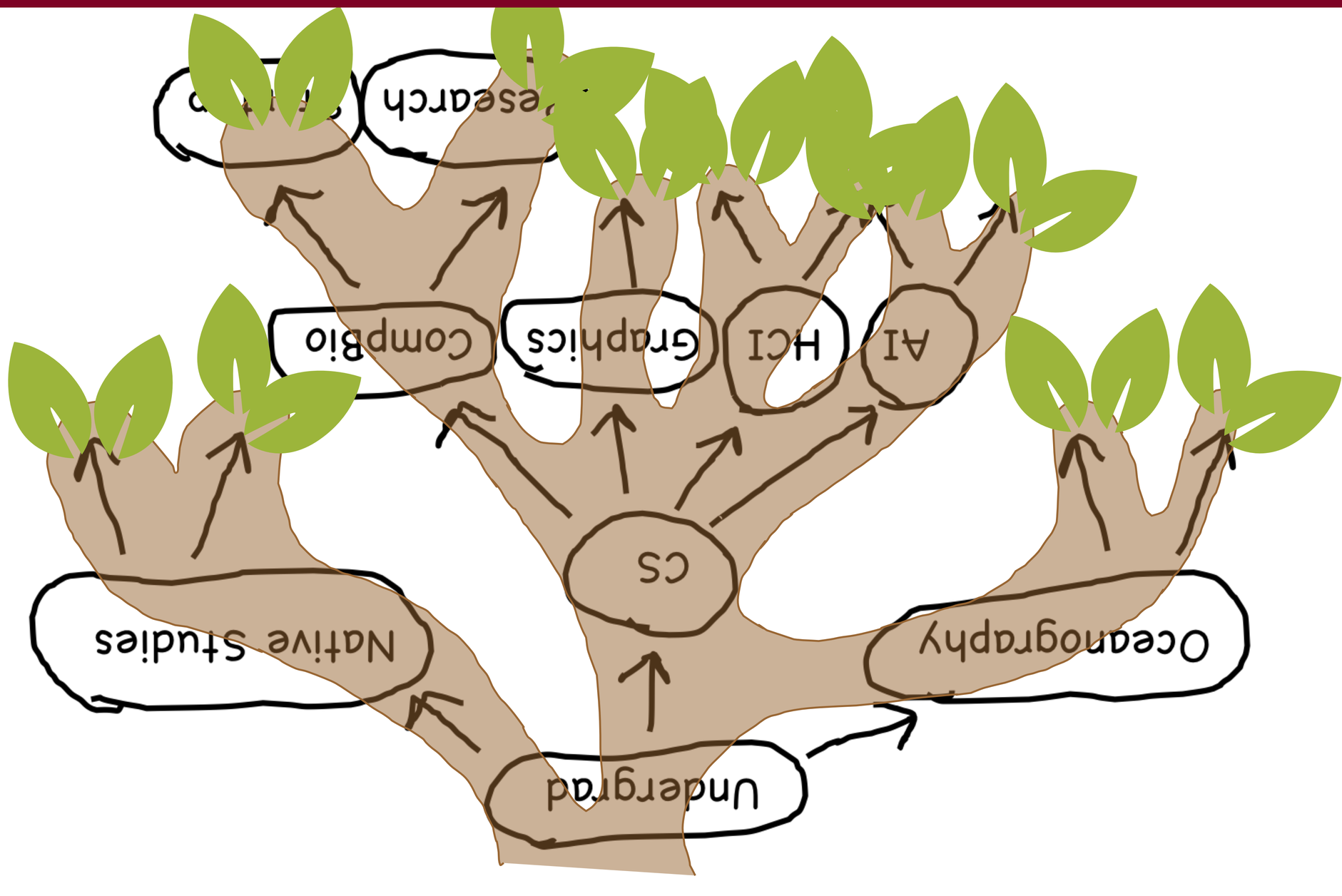
Decision Tree



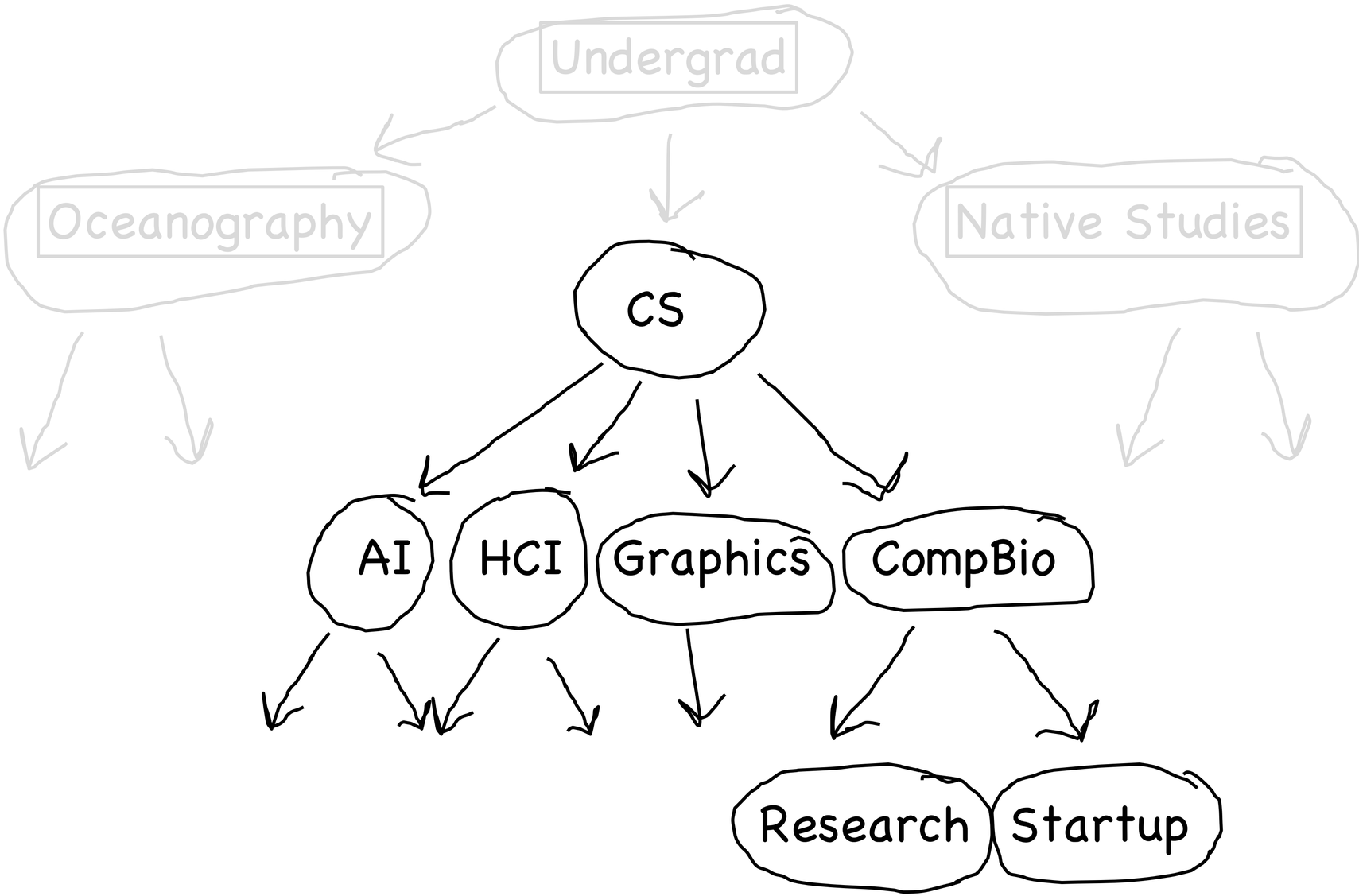
Why Tree?


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

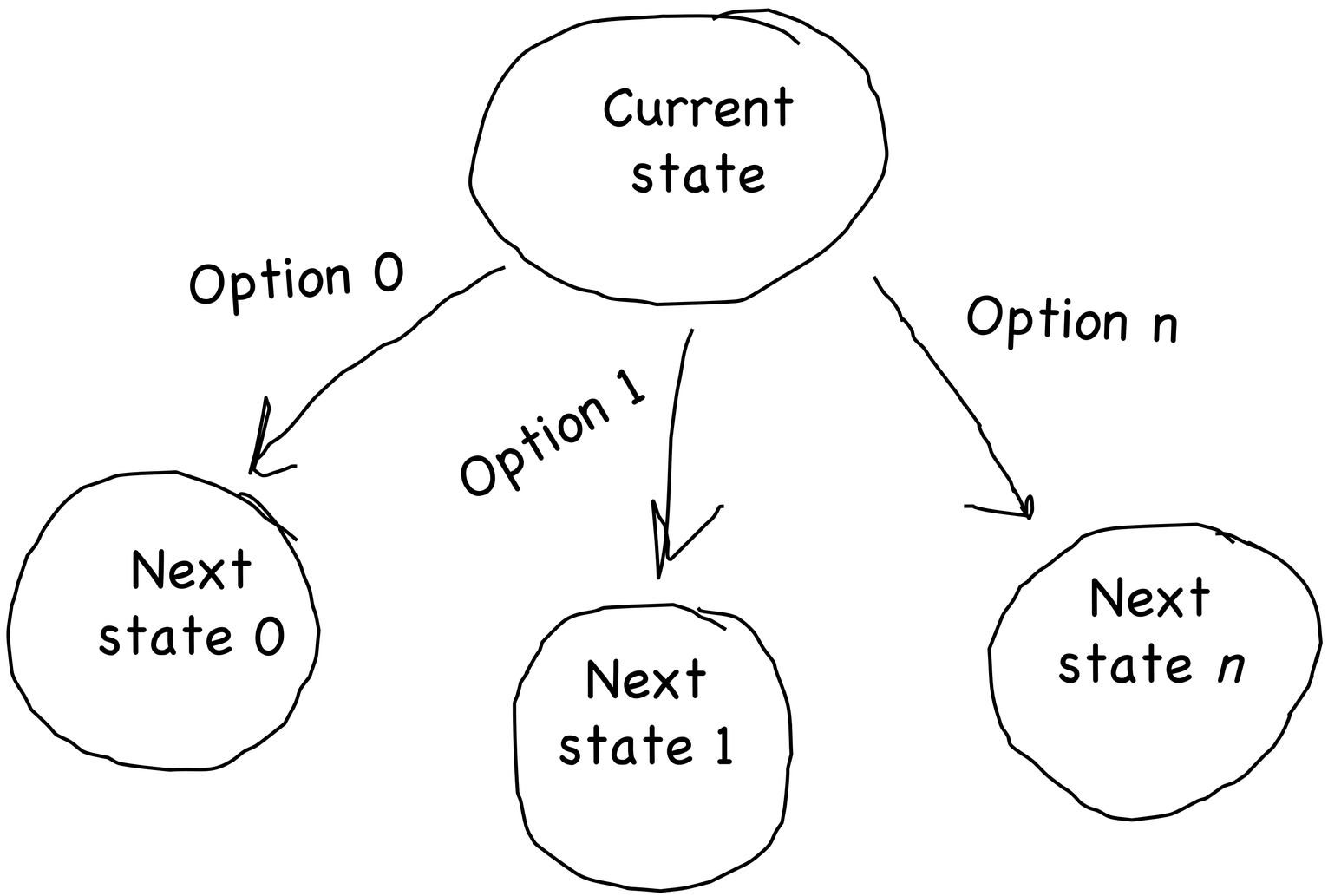
Tree


Undergrad
CS
Native Studies
Oceanography
CompBio
Graphics
HCI
AI
Research

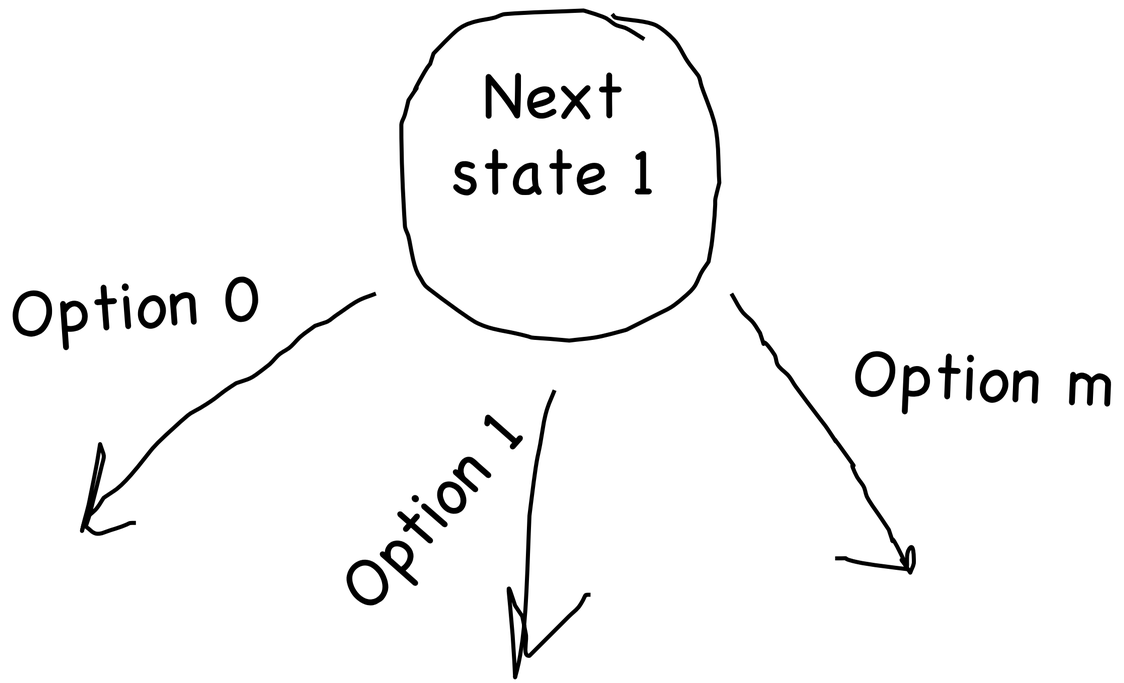
Decision Tree



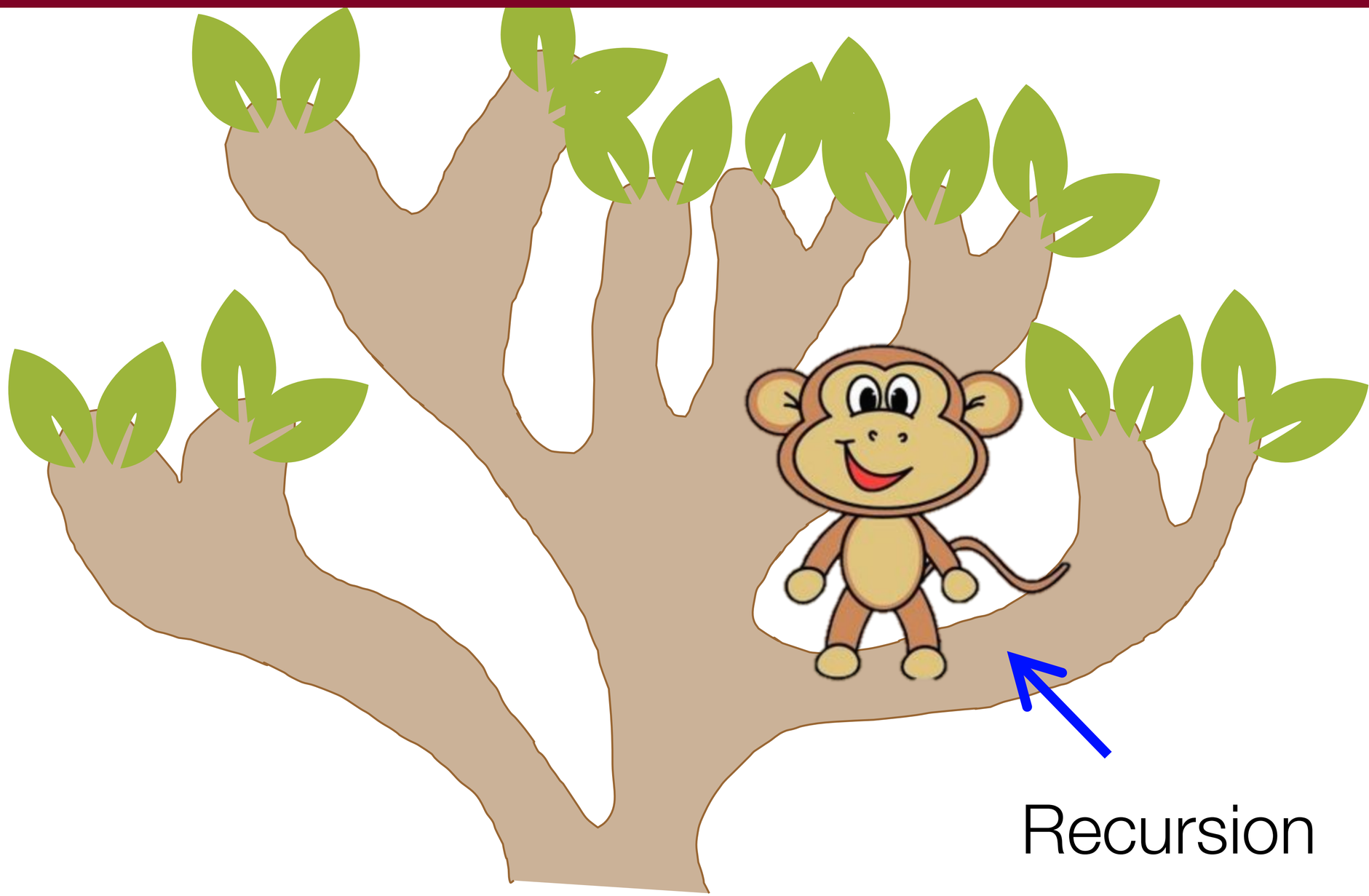
Decision Tree



Decision Tree

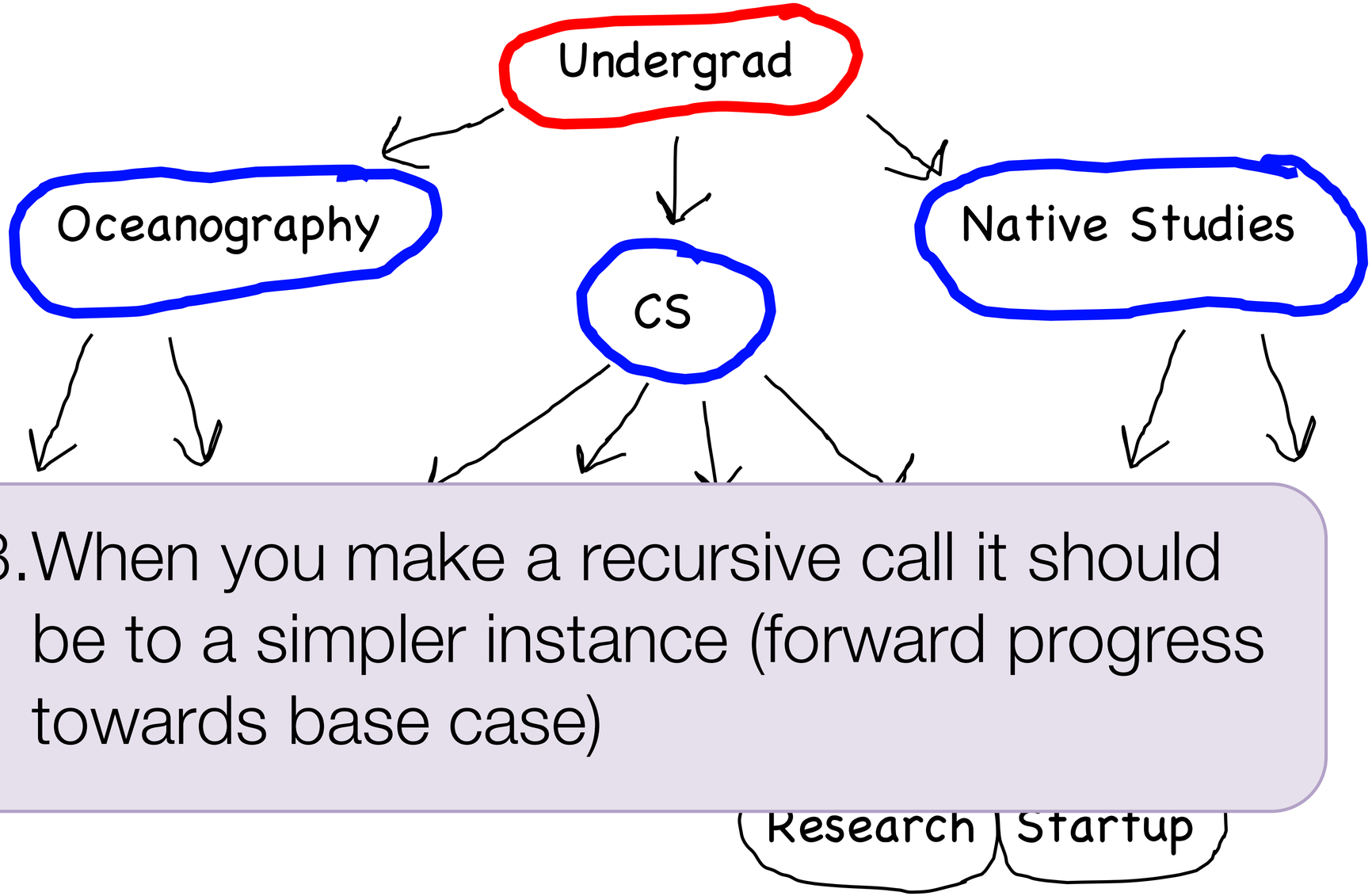


Recursion Hearts Trees



Recursion

outputAllEndStates("undegrad")?



3. When you make a recursive call it should be to a simpler instance (forward progress towards base case)

```
Set<string> next = getNextStates(currState);
```

Recursion on Trees

```
void outputAllStates(string currState) {
  if(isEndState(currState)) {
    cout << currState << endl;
  } else {
    Set<string> allNextStates = getNextStates(currState);
    for(string next : allNextStates) {
        outputAllStates(next);
    }
  }
}
```

Recursion on Trees

```
void outputAllStates(string currState) {
  if(isEndState(currState)) {
    cout << currState << endl;
  } else {
    Set<string> allNextStates = getNextStates(currState);
    for(string next : allNextStates) {
        outputAllStates(next);
    }
  }
}
```

“undergrad”

allNextStates:

[

“Oceanography”,

“CS”

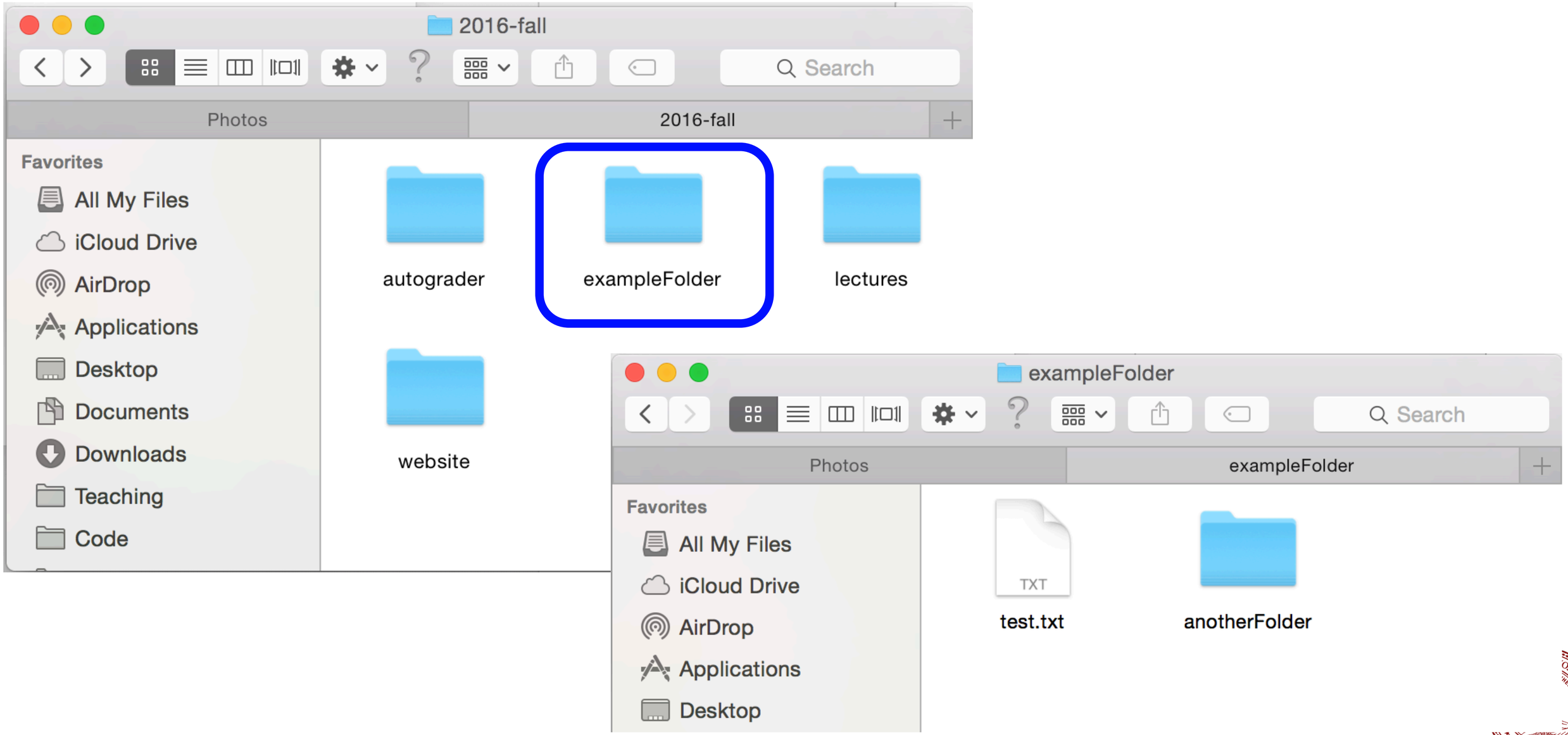
“Native Studies”

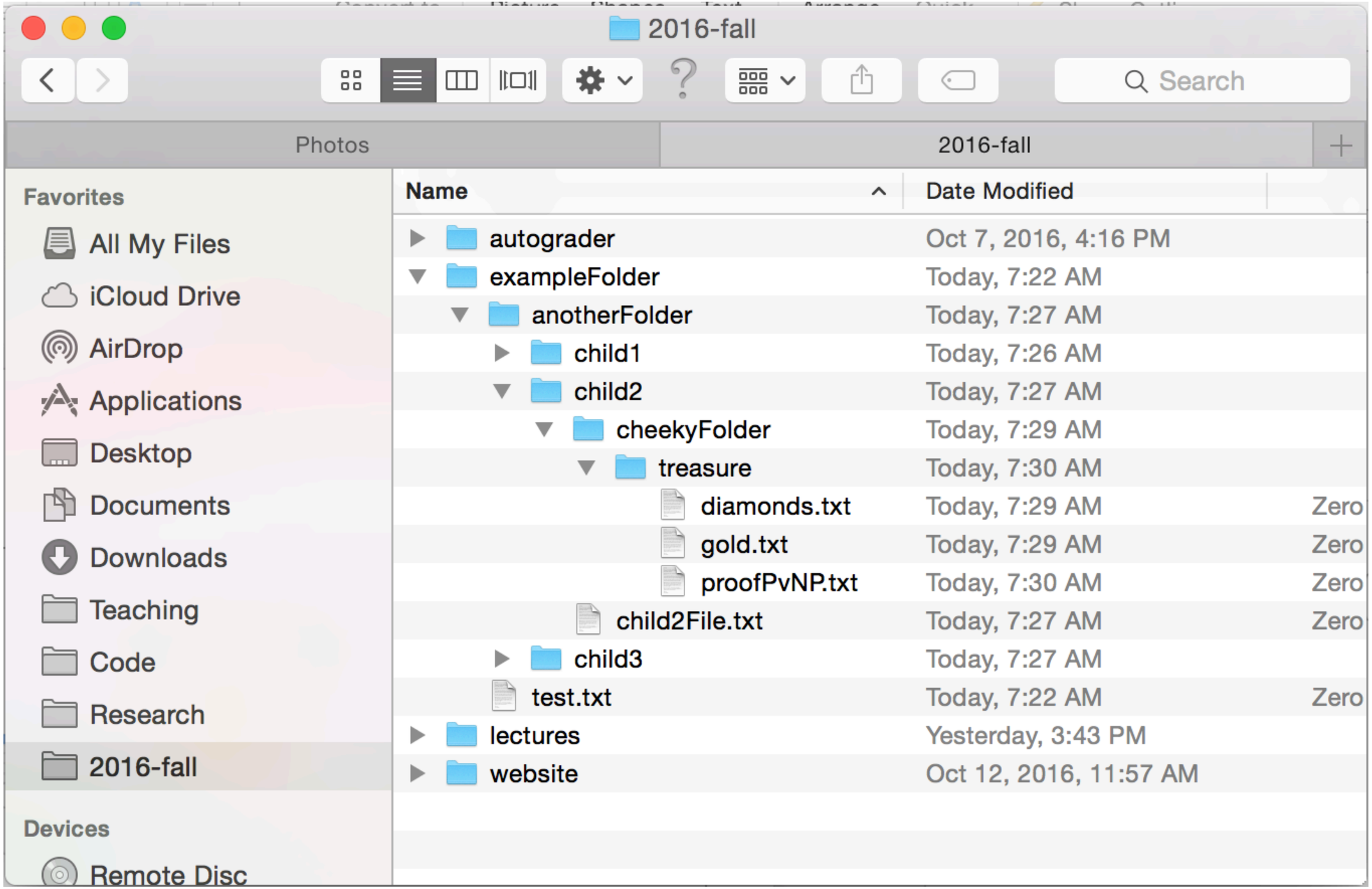
]

next = “Oceanography”

The point: get the right metaphor in your head.

List all Paths in a Folder

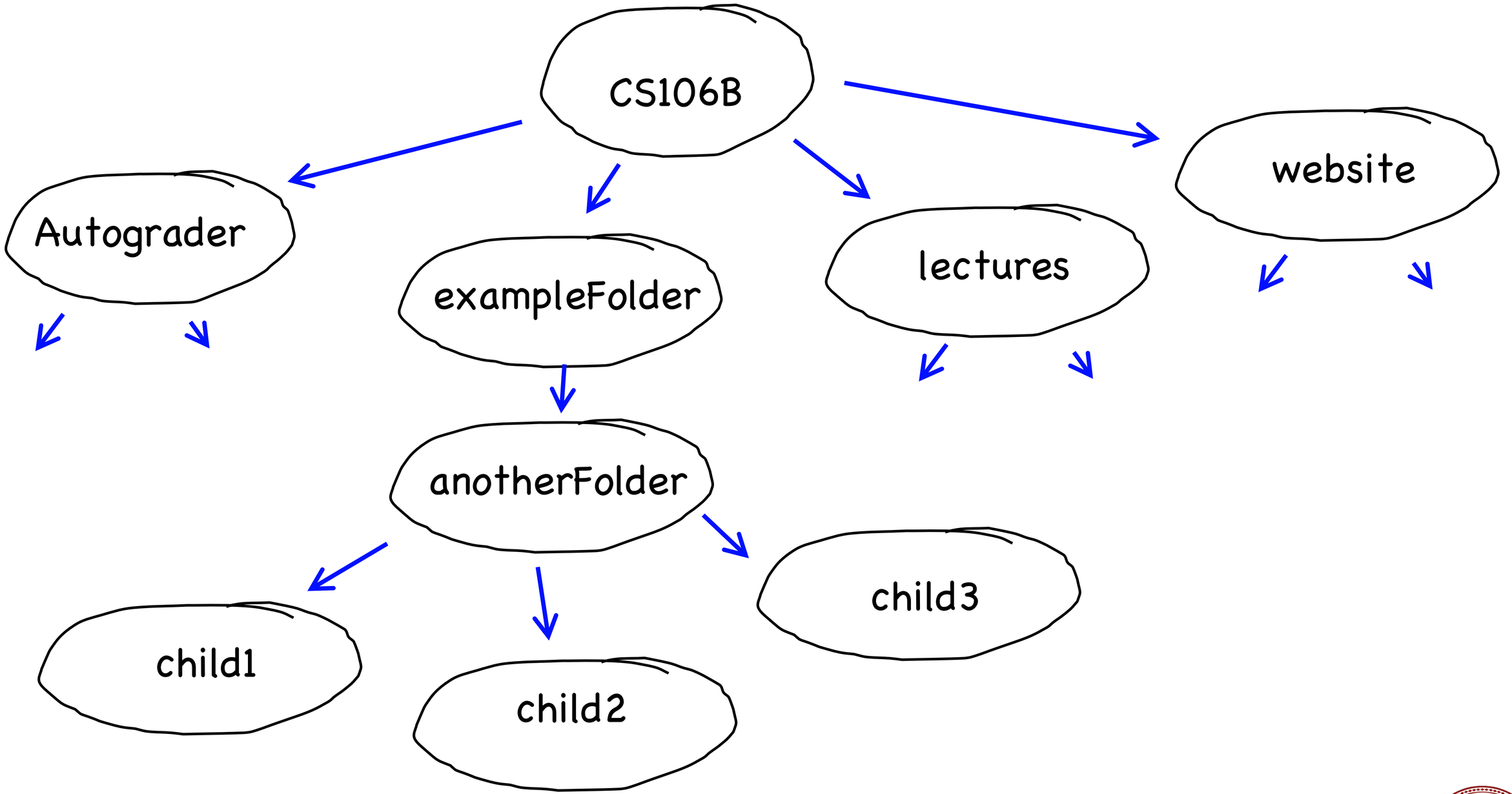


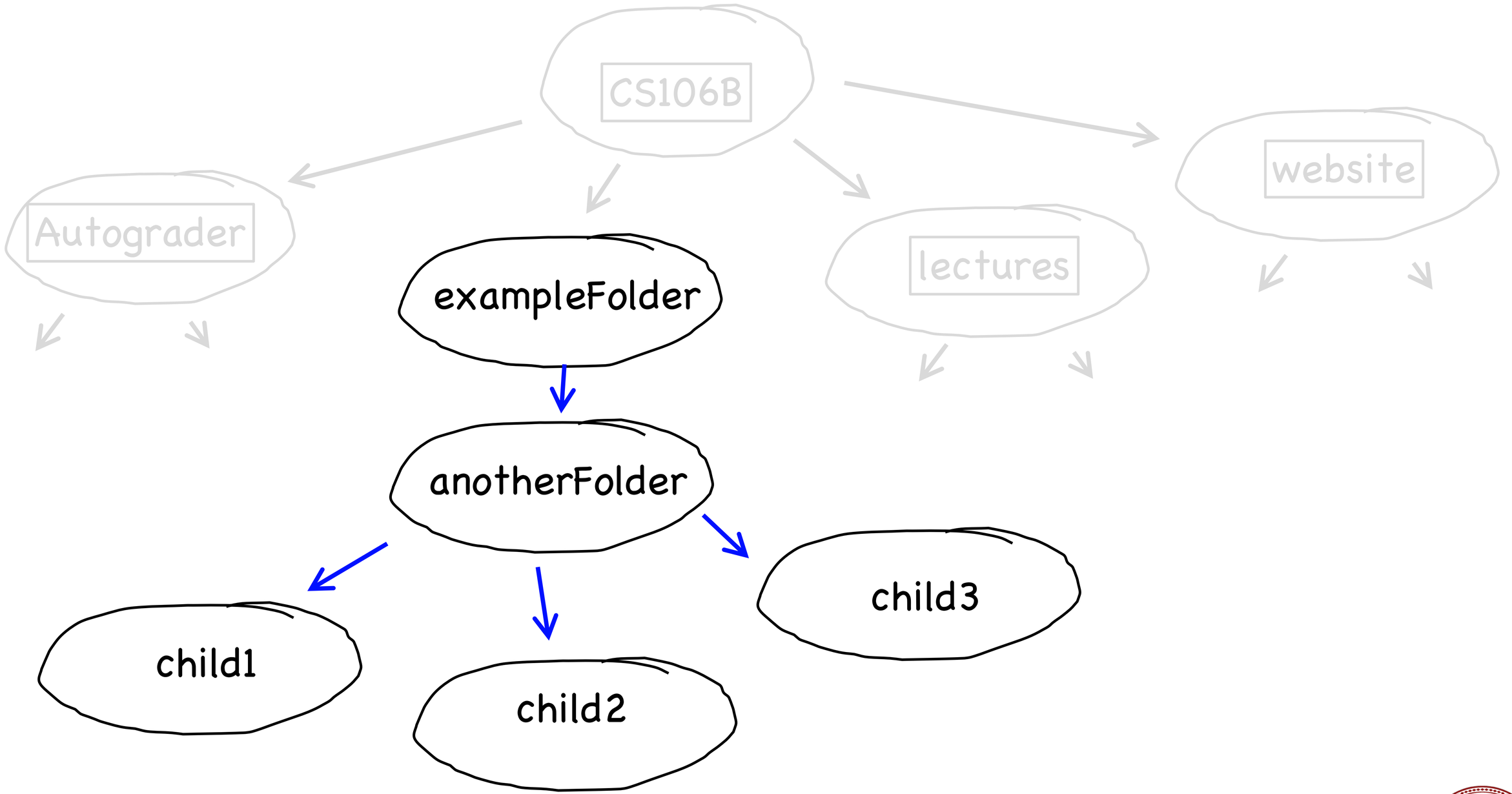

2016-fall
Search
Photos
2016-fall
Favorites
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Teaching
Code
Research
2016-fall
Devices
Remote Disc
Name
Date Modified
autograder
Oct 7, 2016, 4:16 PM
exampleFolder
Today, 7:22 AM
anotherFolder
Today, 7:27 AM
child1
Today, 7:26 AM
child2
Today, 7:27 AM
cheekyFolder
Today, 7:29 AM
treasure
Today, 7:30 AM
diamonds.txt
Today, 7:29 AM
Zero
gold.txt
Today, 7:29 AM
Zero
proofPvNP.txt
Today, 7:30 AM
Zero
child2File.txt
Today, 7:27 AM
Zero
child3
Today, 7:27 AM
test.txt
Today, 7:22 AM
Zero
lectures
Yesterday, 3:43 PM
website
Oct 12, 2016, 11:57 AM

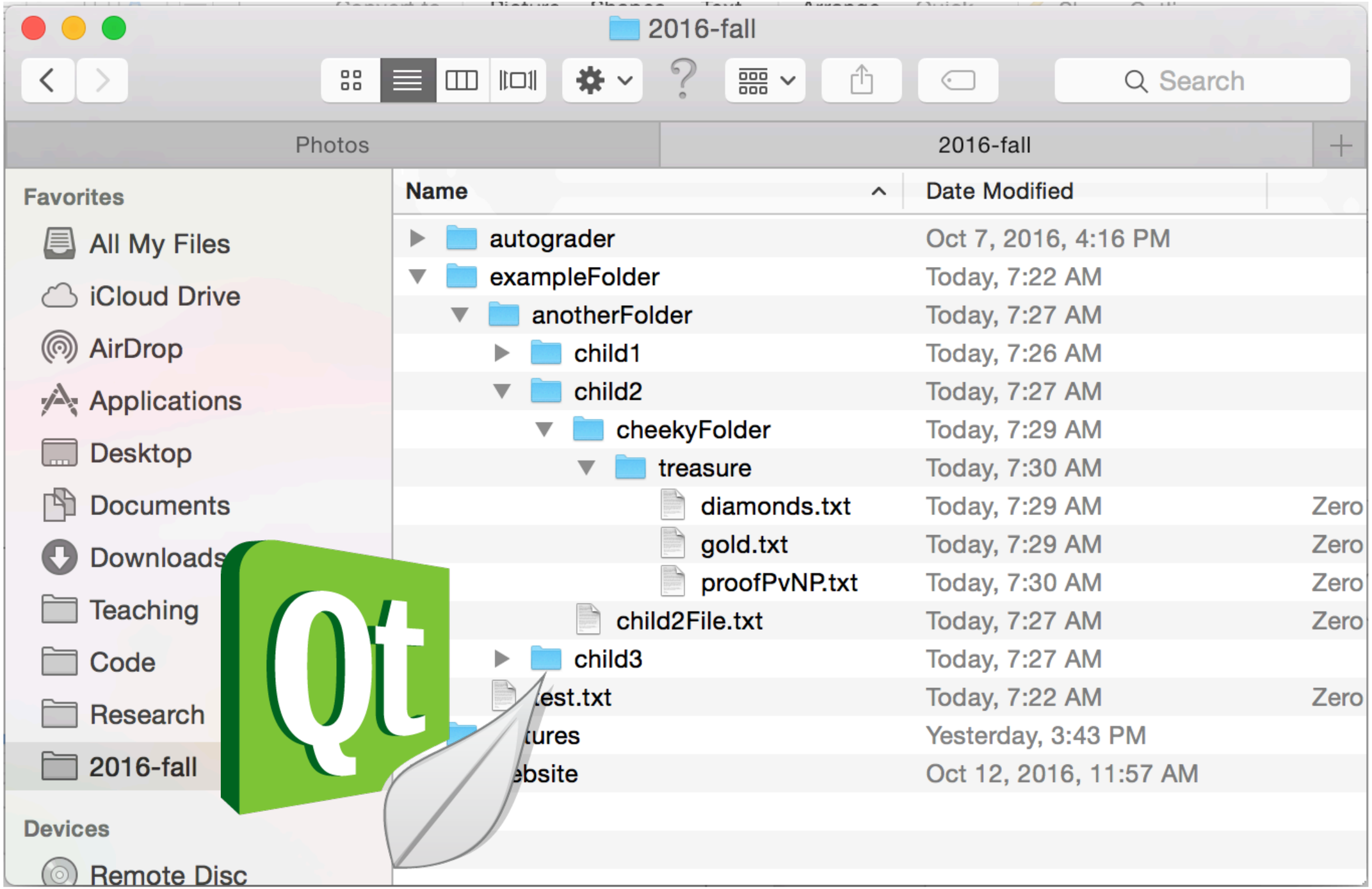
A Folder Is a Recursive Container

A folder (aka directory) is a file system container that can hold files and other folders.

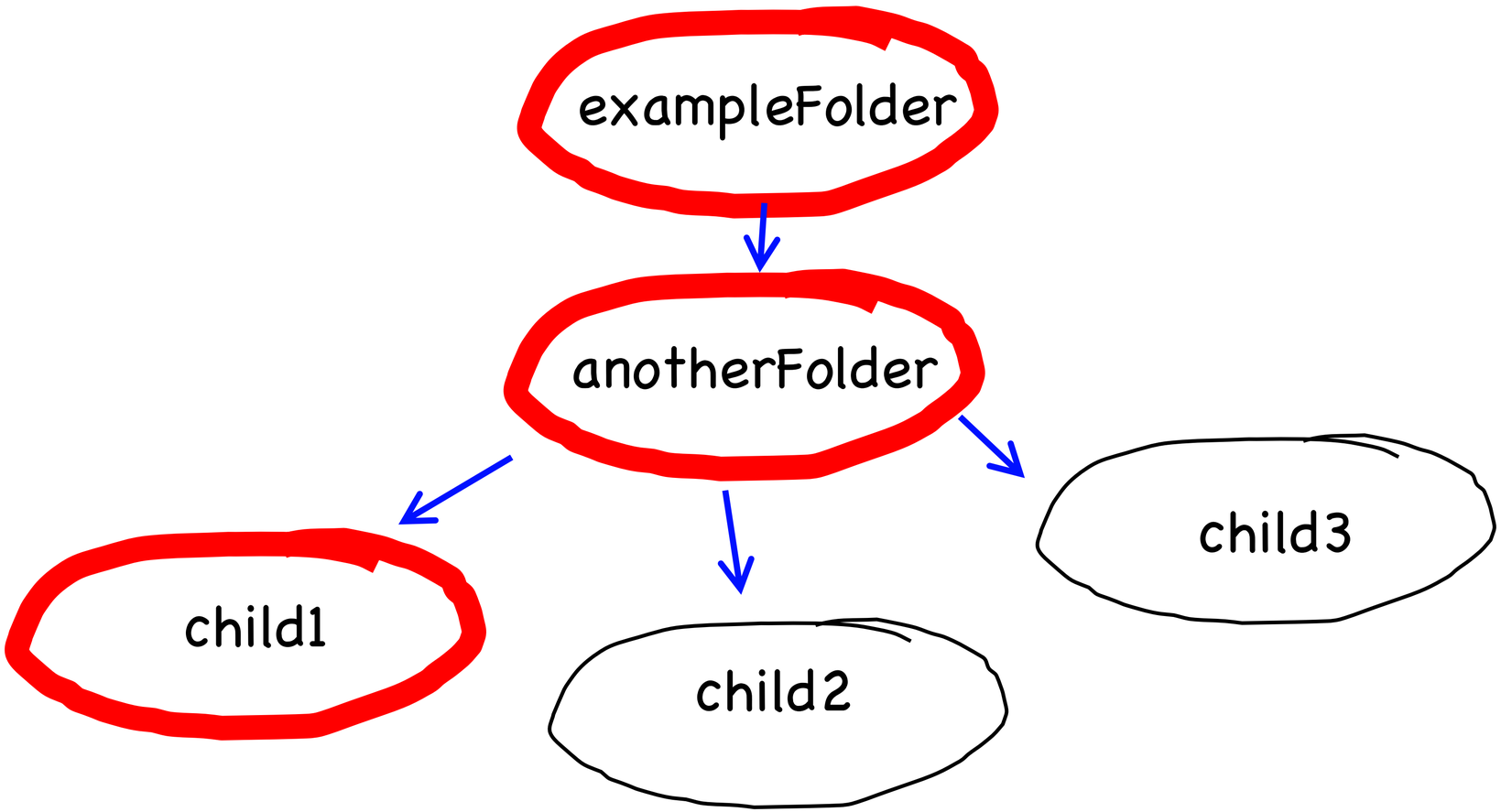
A folder is a tree!


CS106B
Autograder
exampleFolder
lectures
website
anotherFolder
child1
child2
child3


CS106B
Autograder
website
lectures
exampleFolder
anotherFolder
child1
child2
child3


2016-fall
Search
Photos
2016-fall
Favorites
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Teaching
Code
Research
2016-fall
Devices
Remote Disc
Name
Date Modified
autograder
Oct 7, 2016, 4:16 PM
exampleFolder
Today, 7:22 AM
anotherFolder
Today, 7:27 AM
child1
Today, 7:26 AM
child2
Today, 7:27 AM
cheekyFolder
Today, 7:29 AM
treasure
Today, 7:30 AM
diamonds.txt
Today, 7:29 AM
Zero
gold.txt
Today, 7:29 AM
Zero
proofPvNP.txt
Today, 7:30 AM
Zero
child2File.txt
Today, 7:27 AM
Zero
child3
Today, 7:27 AM
Today, 7:22 AM
Zero
Yesterday, 3:43 PM
Oct 12, 2016, 11:57 AM
Qt

Visualize the Recursion



A Little Word Puzzle

“What nine-letter word can be reduced to a single-letter word one letter at a time by removing letters, leaving it a legal word at each step?”

The Startling Truth

S	T	A	R	T	L	I	N	G

Is there **really** just one nine-letter word with this property?

Iterative?

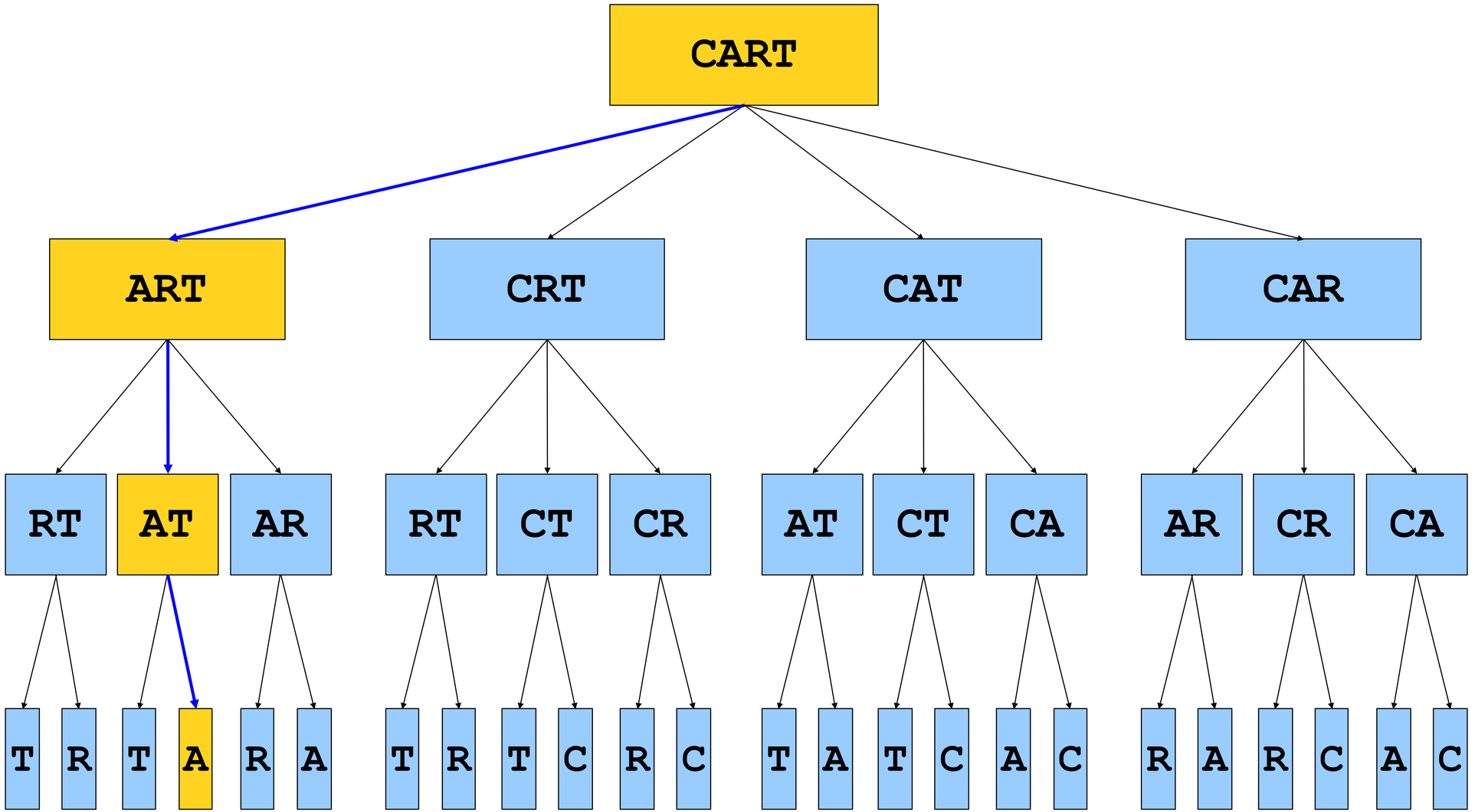

LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Recursive?


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Decision Tree?

Reduce Decision Tree



Decision Tree Search

```
bool search(currentState) {
  if(isSolution(currentState)) {
    return true;
  } else {
    for(option : moves from currentState) {
      nextState = takeOption(curr, option);
      if(search(nextState)){
        return true;
      }
    }
    return false;
  }
}
```

Reducible Word

Let's define a **reducible word** as a word that can be reduced down to one letter by removing one character at a time, leaving a word at each step.

- **Base Cases:**
  - The empty string
- **Recursive Step:**
  - Any multi-letter word is reducible if you can remove a letter (legal move) to form a shrinkable word.

Is there **really** just one nine-letter word with this property?

Reducible Word

```
bool reducible(Lexicon & lex, string word) {
  if(word.length()==1 && lex.contains(word)){
    return true;
  } else {
    for(int i=0; i < word.length(); i++) {
      string copy = word;
      copy.erase(i, 1);
      if(lex.contains(copy)){
        if(reducible(lex, copy)){
          return true;
        }
      }
    }
    return false;
  }
}
```

Get all legal moves, and corresponding next states

Reducible Word

```
bool reducable(Lexicon & lex, string word) {
  if(word.length()==1 && lex.contains(word)){
    return true;
  } else {
    for(int i=0; i < word.length(); i++) {
      string copy = word;
      copy.erase(i, 1);
      if(lex.contains(copy)){
        if(reducable(lex, copy)){
          return true;
        }
      }
    }
    return false;
  }
}
```

If any decision is reducible return true

Reducible Word

```
bool reducable(Lexicon & lex, string word) {
  if(word.length()==1 && lex.contains(word)){
    return true;
  } else {
    for(int i=0; i < word.length(); i++) {
      string copy = word;
      copy.erase(i, 1);
      if(lex.contains(copy)){
        if(reducable(lex, copy)){
          return true;
        }
      }
    }
    return false;
  }
}
```

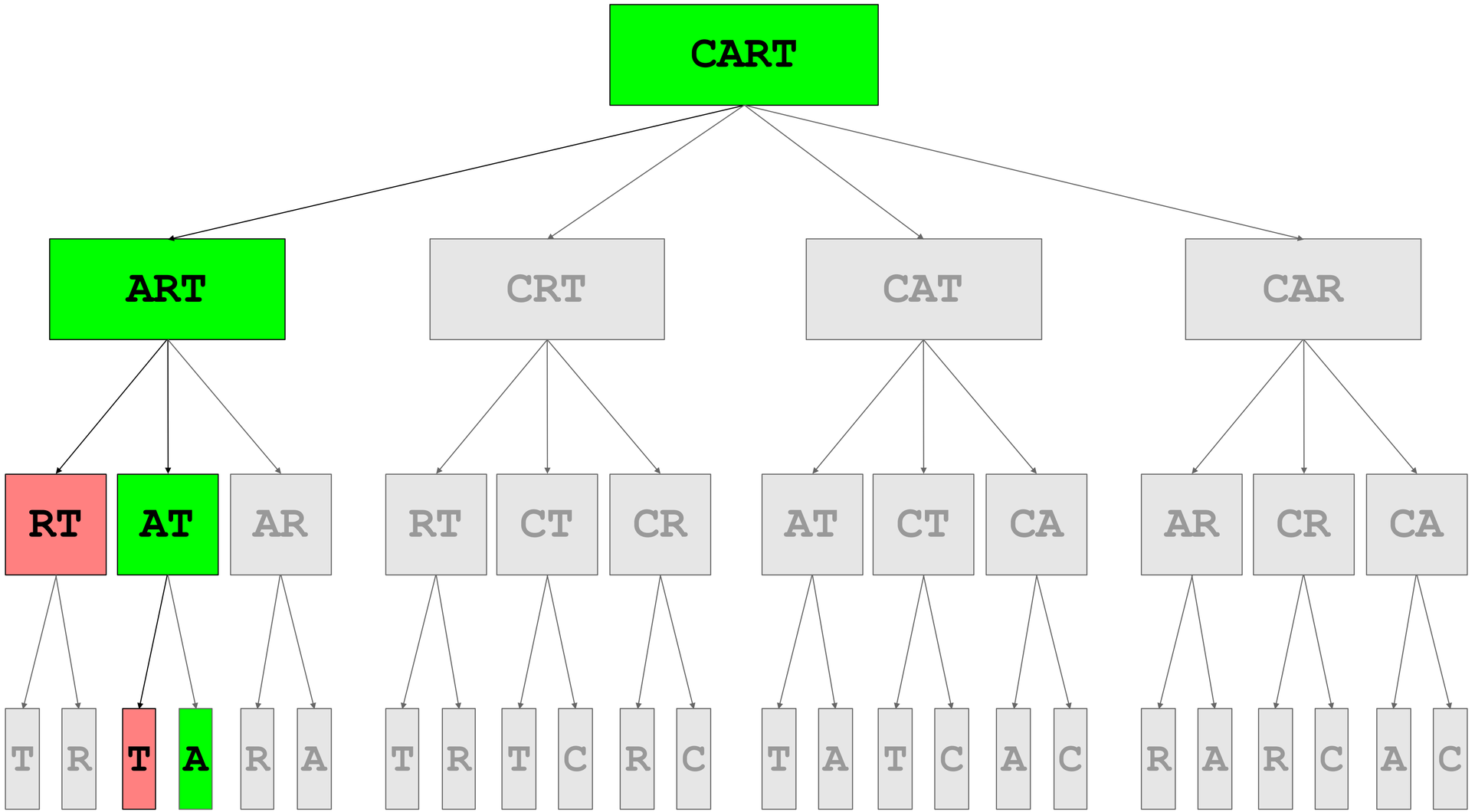
Only return false if every single option failed.

Recursive Exploration of Tree

The function we have just defined is an example of **recursive exploration of a tree**. In this case we are looking for any path through the decision tree. For a given state:

- If *any* option leads to succeeds, that's great! We're done.
- If *none* of the options succeed, then this particular problem can't be solved from the state.

Visualize the Recursion



Tree Search

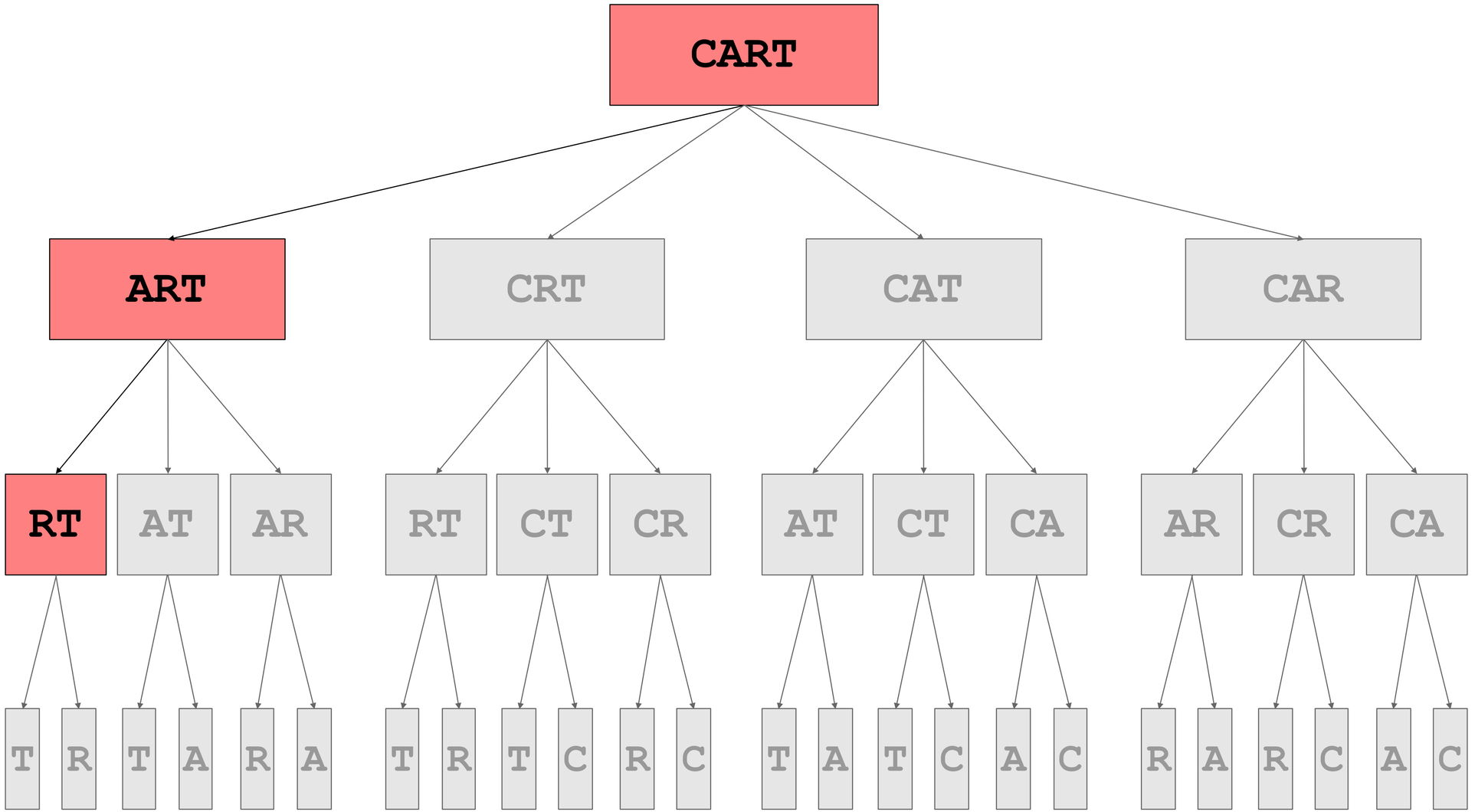
```
bool reducible(Lexicon & lex, string word) {
  if(word.length()==1 && lex.contains(word)){
    return true;
  } else {
    for(int i=0; i < word.length(); i++) {
      string copy = word;
      copy.erase(i, 1);
      if(lex.contains(copy)){
        if(reducible(lex, copy)){
          return true;
        }
      }
    }
    return false;
  }
}
```

Ur Doin It Rong!

```
bool reducible(Lexicon & lex, string word) {
  if(word.length()==1 && lex.contains(word)){
    return true;
  } else {
    for(int i=0; i < word.length(); i++) {
      string copy = word;
      copy.erase(i, 1);
      if(lex.contains(copy)){
        if(!reducible(lex, copy)){
          return false;
        }
      }
    }
    return true;
  }
}
```

Note how the true became a false

Ur Doin It Rong!



Generating All Permutations

Generate all permutations of a string

`Eg: “abcd”`

`Eg: “abcdefghijklmnop”`

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

What is the Permutation Tree?

Problems with the Same Underlying Form

1. List all **permutations** of a set of elements
2. Output the path of all files in a **folder**.
3. Find all words that are "**reducible**"
4. Play a game of knights tour **solitaire**
5. Find the optimal "alignment" of two strands of **DNA**

They have a similar underlying representation
and they can all be solved with recursion

Your Brain is Recursive

You Can Imagine Many Next Steps

You Can Use the Same “Function” To Imagine Further Futures

Executes the same “function” with different inputs

Corballis, M.C. The Recursive Mind. Princeton University Press, 2011.

Happy Friday

Knight’s Tour

35	40	47	44	61	08	15	12
46	43	36	41	14	11	62	09
39	34	45	48	07	60	13	16
50	55	42	37	22	17	10	63
33	38	49	54	59	06	23	18
56	51	28	31	26	21		03
29	32	53	58	05	02	19	24
52	57	30	27	20	25	04	01

Knight's Tour Demo

Knight's Tour