

CompSci 101

Introduction to Computer Science



Nov 28, 2017

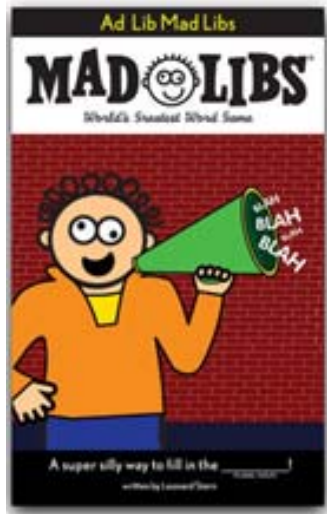
Prof. Rodger

Announcements

- Last RQ, RQ18 Thursday!
- Assign 8 due Dec 5, Assign 9 due Dec 8
- APT 8 Due Dec 7
- Exam 2 back – Regrades through Dec 7
- Lab this week
- Today:
 - Assign 8 - Recommender
 - How do you access directories/folders
 - Recursion – Solving problems by solving smaller and smaller similar problems

Exam 2 Back

- Stats
 - Average 85/92
- Request regrade through gradescope by Dec 7
- Questions?



Lab this week!

MAD  **TAKES**

DRAGONS AD-LIB

Contributed by SAM



Printer Friendly

COLOR	Purple	
SUPERLATIVE (ENDING IS "EST")	best	
ADJECTIVE	confident	Random Word Totally Random
BODY PART (PLURAL)	knees	Random Word
BODY PART	heart	Random Word
NOUN	cotton	Random Word
ANIMAL (PLURAL)	Turtles	Random Word
ADJECTIVE	kind	Random Word
ADJECTIVE	encouraging	Random Word
ADJECTIVE	obedient	Random Word

Go Mad!

Lab this week (cont)

DRAGONS AD-LIB

Contributed by *SAM*



Printer Friendly

The Purple Dragon is the Best Dragon of all. It has Confident Knees, and a Heart shaped like a Cotton. It loves to eat Turtles, although it will feast on nearly anything. It is Kind and Encouraging. You must be Obedient around it, or you may end up as it's meal!

Math, Engineering, Sociology

- Netflix prize in 2009
 - Beat the system, win
 - <http://nyti.ms/sPvR>



Assignment 8: Collaborative Filtering

- How does Amazon know what I want?
 - Lots of customers, lots of purchases
- How does Pandora know music like Kanye's?
 - This isn't really collaborative filtering, more content-based
- How does Netflix recommend movies?
 - Why did they offer one million \$\$ to better their method?
- Students at Duke who like Compsci also like ...
 - Could this system be built?

From User Rating to Recommendations



Spectre	Martian	Southpaw	Everest	PitchPerfect 2
3	-3	5	-2	-3
2	2	3	2	3
4	4	-2	1	-1

- | **What should I choose to see?**
 - What does this depend on?
- | **Who is most like me?**
 - How do we figure this out

ReadAllFood modules: Food Format

bit.ly/101f17-1128-1

- All Reader modules return a tuple of strings: itemlist and dictratings dictionary

```
Sarah Lee  
(DivinityCafe) (3)  
(IlForno) (3)  
(TheSkillet) (-3)  
(LoopPizzaGrill) (3)  
(FarmStead) (3)  
(Tandoor) (5)  
(PandaExpress) (-3)
```

```
Melanie  
(McDonalds) (1)  
(Tandoor) (3)  
(DivinityCafe) (5)  
(TheCommons) (3)  
(TheSkillet) (1)  
(IlForno) (3)  
(PandaExpress) (3)  
J J  
(TheSkillet) (1)
```

not
all
shown
...

- Translated to list and dictionary:

```
['IlForno', 'TheCommons', 'FarmStead', 'DivinityCafe', 'PandaExpress',  
'TheSkillet', 'Tandoor', 'LoopPizzaGrill', 'McDonalds']
```

```
{'Sung-Hoon': [-1, 1, -1, 0, 3, -3, -3, 5, 1], 'Wei': [0, 3, 1, 1, 0, 0, 5,  
3, -1], 'Sly one': [1, 3, 0, 5, 0, 3, 3, 3, 0], 'Nana Grace': [3, 3, 0, 5,  
0, 0, 1, -5, -1], 'Melanie': [3, 3, 0, 5, 3, 1, 3, 0, 1], 'J J': [0, 0, 1,  
0, 1, 1, 3, -1, 1], 'Harry': [0, 5, 3, 5, -5, 1, 0, -1, -3], 'Sarah Lee':  
[3, 0, 3, 3, -3, -3, 5, 3, 0]}
```

Data For Recommender

- Users/Raters rate Items
 - We need to know the items
 - We need to know how users rate each item
- Which eatery has highest average rating?
 - Conceptually: average columns in table
 - How is data provided in this assignment?

	ABP	BlueEx	McDon	Loop	Panda	Nasher
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Data For Recommender

- itemlist are provided in a list of strings
 - Parsing data provides this list
- dictratings provided in dictionary
 - Key is user ID
 - Value is list of integer ratings

	ABP	BlueEx	McDon	Loop	Panda	Nasher
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Data For Recommender

- Given Parameters
 - itemlist: a list of strings
 - dictratings: dictionary of ID to ratings list
- Can you write
 - Average(itemlist, dictratings)

	ABP	BlueEx	McDon	Loop	Panda	Nasher
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Drawbacks of Item Averaging

- Are all ratings the same to me?
 - Shouldn't I value ratings of people "near" me as more meaningful than those "far" from me?
- Collaborative Filtering
 - https://en.wikipedia.org/wiki/Collaborative_filtering
 - How do we determine who is "near" me?
- Mathematically: treat ratings as vectors in an N-dimensional space, $N = \# \text{ ratings}$
 - Informally: assign numbers, higher the number, closer to me

Collaborative Filtering: Recommender

- First determine closeness of all users to me:
 - "Me" is a user-ID, parameter to function
 - Return list of (ID, closeness-#) tuples, sorted
- Use just the ratings of person closest to me
 - Is this a good idea?
 - What about the 10 closest people to me?
- What about weighting ratings
 - Closer to me, more weight given to rating

How do you calculate a similarity?

- Me: $[3, 5, -3]$
 - Joe: $[5, 1, -1]$
 - Sue: $[-1, 1, 3]$
-
- Joe to Me
 - Sue to Me

Collaborative Filtering

- For Chris: $12 * [1, 1, 0, 3, 0, -3] =$
 $- [12, 12, 0, 36, 0, -36]$
- For Sam: $[0, 75, 125, 0, -75, 125]$



	ABP	BlueEx	McDon	Loop	Panda	Nasher
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Adding lists of numbers

[12, 12, 0, 36, 0, -36]

[0, 75, 125, 0, -75, 125]

[-111, 111, 111, 185, 37, -37]

[-99, 198, 236, 221, -38, 52]

- Adding columns in lists of numbers
 - Using indexes 0, 1, 2, ... sum elements of list
 - `sum([val[i] for val in d.values()])`

Then divide by number of nonzeros

[12, 12, 0, 36, 0, -36]

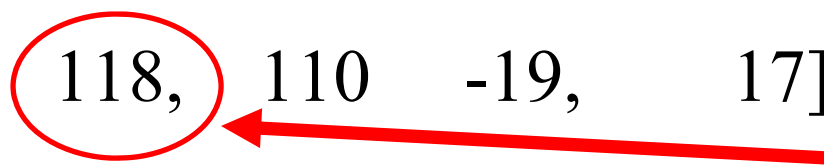
[0, 75, 125, 0, -75, 125]

[-111, 111, 111, 185, 37, -37]

[-99, 198, 236, 221, -38, 52]

/2 /3 /2 /2 /2 /3

[-49, 66, 118, 110, -19, 17]



	ABP	BlueEx	McDon	Loop	Panda	Nasher
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Recommend
3rd item

Follow 12-step process

- ProcessAllFood first!
 - Read input and save it
 - Get list of restaurants – use that ordering! Set?
 - For each person
 - For each restaurant and its rating
 - Must find location of restaurant in itemlist
 - Then update appropriate counter
 - Print any structure you create to check it

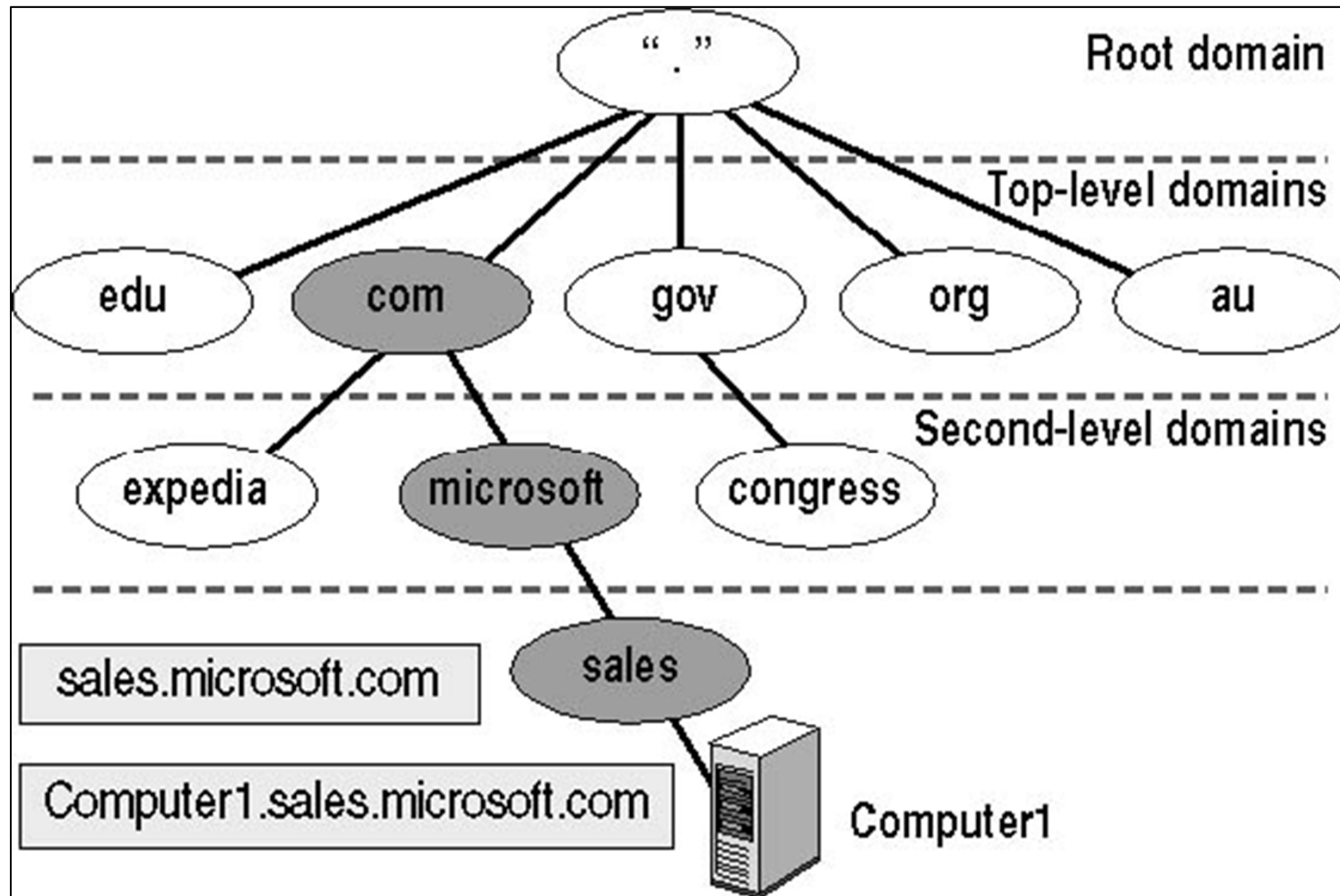
Plan for Today

- Recursion
 - Solving problems by solving similar but smaller problems
- Programming and understanding ...
 - Hierarchical structures and concepts
 - What is a file system on a computer?
 - What is the Internet?
 - How does the Domain Name System Work?
- How do you access directories?
 - And all the files in a directory, and the ...

Recursion

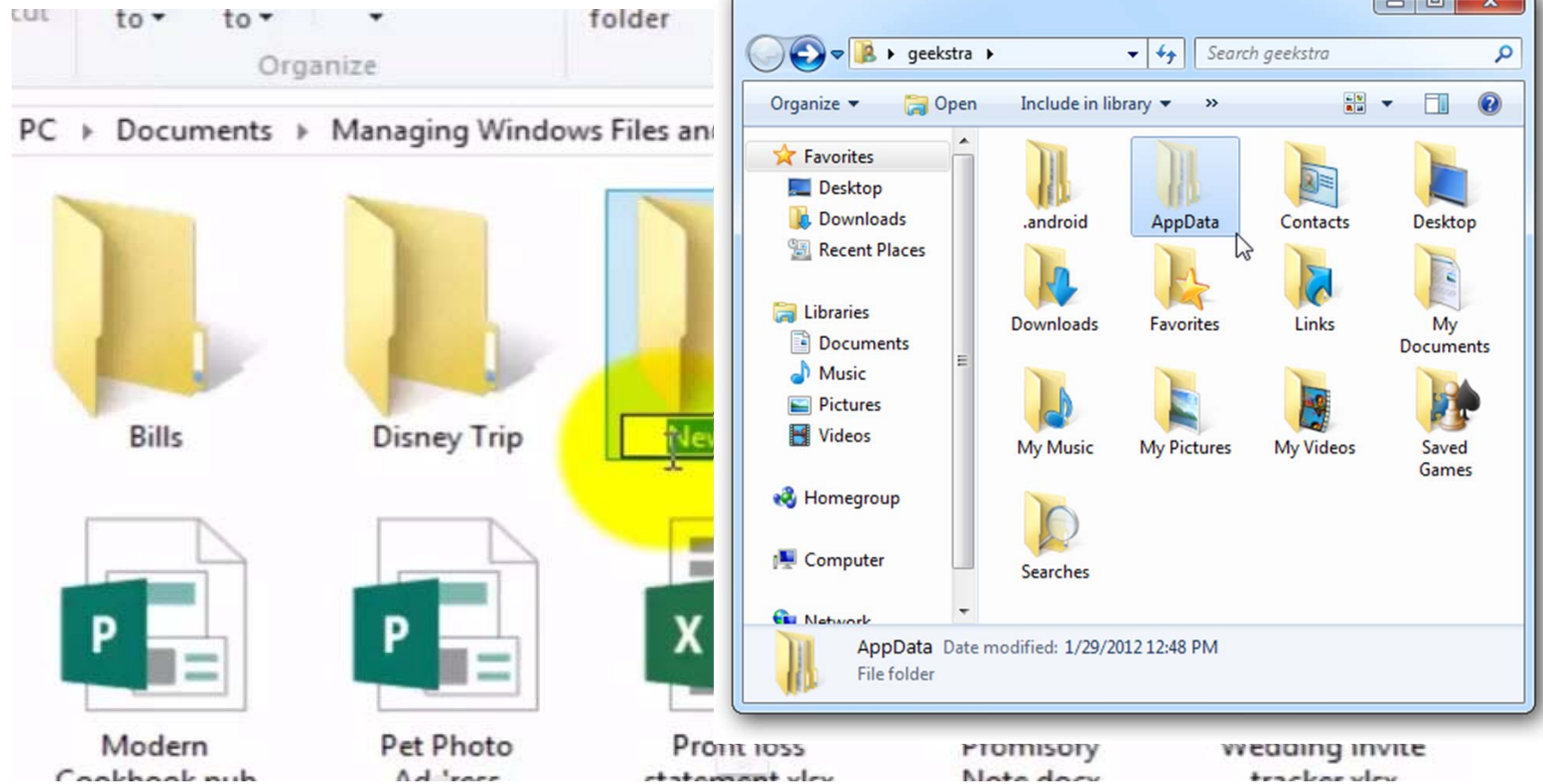
Solving a problem by solving similar but smaller problems

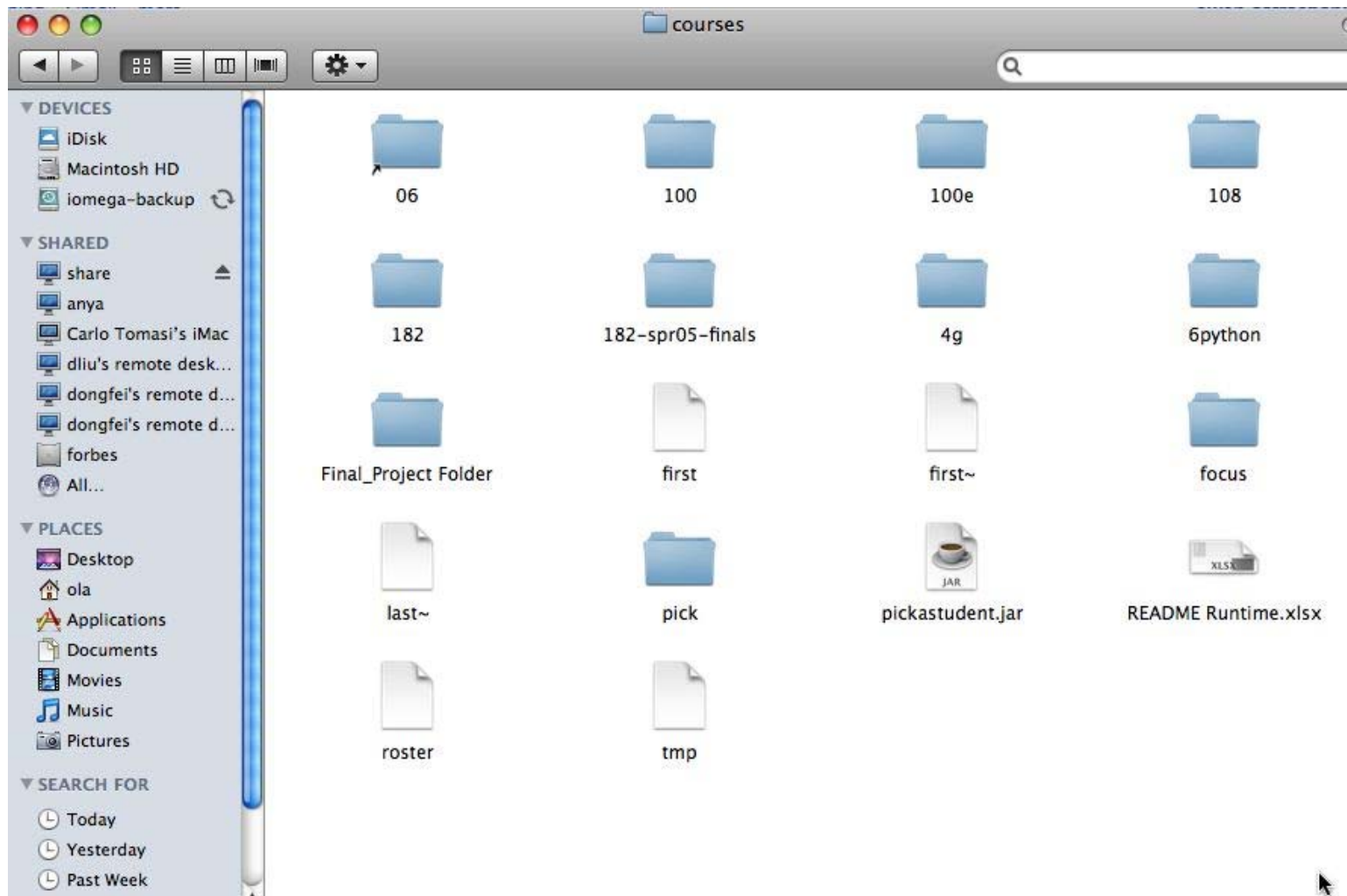
Domain Name System (DNS)



Link: <http://computer1.sales.microsoft.com>

What's in a file-system Folder?

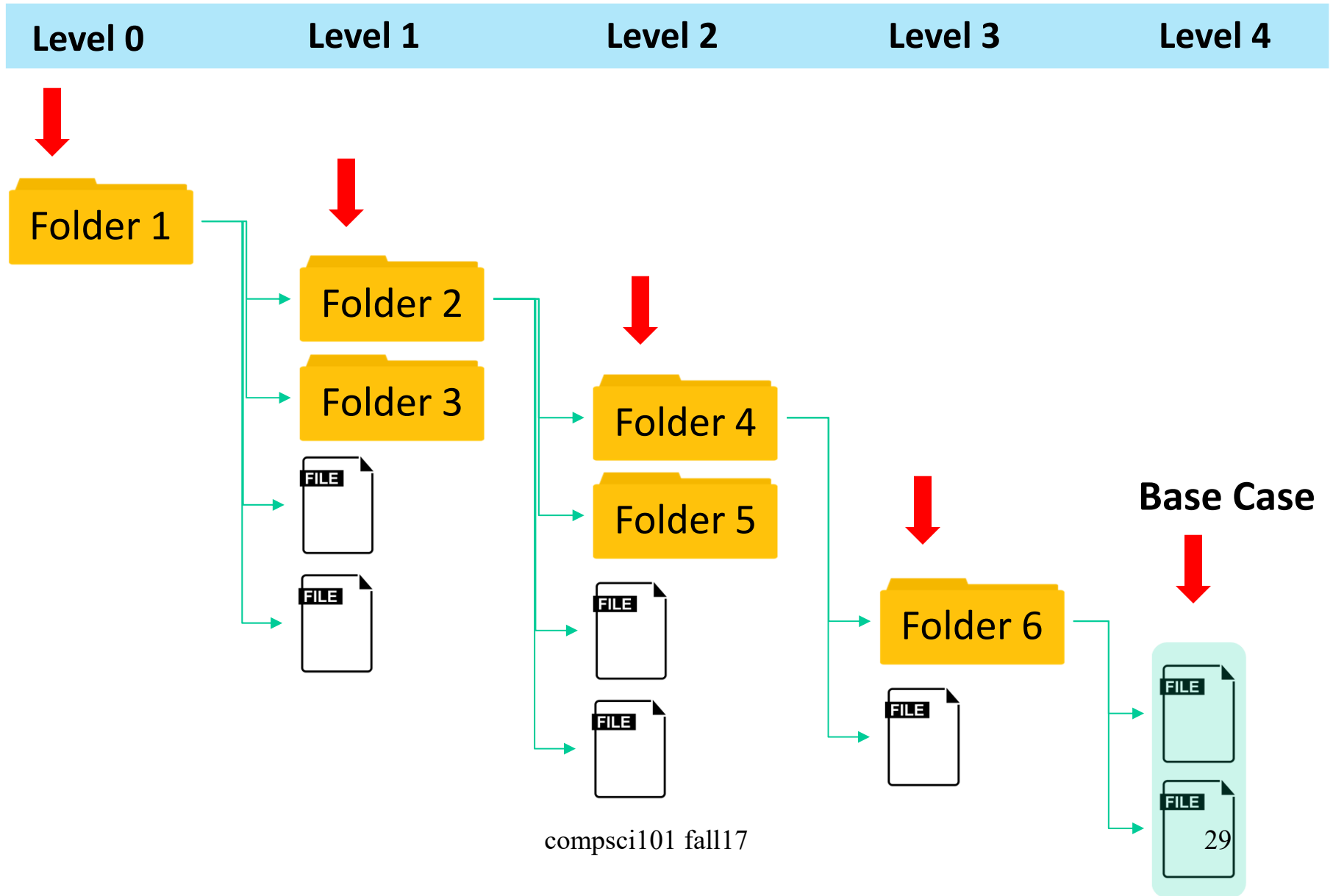




What's in a folder on your computer?

- Where are the **large** files?
- How do you **find them**?
- They take up space!
 - What's the plan –
 1. Erase?
 2. Backup?

Hierarchy in Folder Structure



Recursion to find ALL files in a folder

- A folder can have sub folders and files
- A file cannot have sub files

```
def visit(dirname):  
    for inner in dirname:  
        if isdir(inner):  Is that a directory?  
            visit(inner)  
        else:  If not a directory, it will be a file  
            print name(inner), size(inner)
```

Finding large files: FileVisit.py

```
def bigfiles(dirname,min_size):
    large = []
    for sub in os.listdir(dirname):
        path = os.path.join(dirname,sub)
        if os.path.isdir(path):
            subs = bigfiles(path,min_size)
            large.extend(subs)
        else:
            size = os.path.getsize(path)
            if size > min_size:
                large.append((path,size))
    return large

# on Mac like this:
#big = bigfiles("/Users/Susan/Documents",10000)
# on Windows like this:
big = bigfiles("C:\\Users\\Susan\\Documents",10000)
```

Example Run

- ('C:\\Users\\Susan\\files\\courses\\cps101\\workspace\\spring2015\\assign4_transform\\data\\romeo.txt', 153088L)
- ('C:\\Users\\Susan\\files\\courses\\cps101\\workspace\\spring2015\\assign4_transform\\data\\twain.txt', 13421L)
- ('C:\\Users\\Susan\\files\\courses\\cps101\\workspace\\spring2015\\assign5_hangman\\src\\lowerwords.txt', 408679L)
- ...

Finding Large Files questions

bit.ly/101f17-1128-2

The os and os.path libraries

- Libraries use an API to isolate system dependencies
 - C:\\x\\y **# windows**
 - /Users/Susan/Desktop **# mac**
- FAT-32, ReFS, WinFS, HFS, HSF+, fs
 - Underneath, these systems are different
 - Python API insulates and protects programmer
- Why do we have `os.path.join(x, y)`?
 - `x = /Users/Susan/Documents`
 - `y = file1.txt`
 - `Output = /Users/Susan/Documents/file1.txt`

Dissecting FileVisit.py

- How do we find the contents of a folder?
 - Another name for folder: directory
- How do we identify folder? (by name)
 - `os.listdir(dirname)` returns a list of files and folder
- Path is `c:\user\ola\foo` or `/Users/ola/bar`
 - `os.path.join(dir,sub)` returns full path
 - Platform independent paths
- What's the difference between file and folder?
 - `os.path.isdir()` and `os.path.getsize()`

Does the function call itself? No!

```
def visit(dirname):  
    for inner in dirname:  
        if isdir(inner):  
            visit(inner)  
        else:  
            print name(inner), size(inner)
```

- Is a file inside itself? No!
- Does pseudo code make sense?
 - Details make this a little harder in Python, but close!

Structure matches Code

Find large files

If you see a folder,

1. Find the large files and subfolders
2. For the subfolders, repeat the process of finding large files and any other folders within that subfolder
3. Repeat the process until you reach the last folder

Compress or Zip a folder

If you see a folder,

1. Find the files and subfolders
2. For the subfolders, repeat the process of finding files and any other folders within that subfolder
3. At the last stage, start compressing files and move up the folder hierarchy

Structure matches Code

- Structure of list of lists
 - Can also lead to processing a list which requires processing a list which ...

[[[a, b], [c, d], [a, [b, c], d]]
(a * (b + c (d + e*f)) + (a* (b+d)))

Recursion

- Simpler or smaller calls
- Must have a base case when no recursive call can be made
 - Example - The last folder in the folder hierarchy will not have any subfolders. It can only have files. That forms the base case

Sir Anthony (Tony) Hoare

There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.



Turing Award, didn't get recursion.....

Inventor of quicksort