How Dictionaries work, Recursion
Live Lecture
W is for …

• World Wide Web
  • Where http meets tcp/ip?

• WiFi
  • We need and use this every day

• Windows
  • From OS to …
x is for …

• XOR
  • (a or b) and not (a and b), a.k.a. symmetric difference

• XML
  • eXtensible Markup Language

• Xerox Parc
  • From Mice to Windows
“For the most part, algorithms didn’t create inequity and inequality, but the fact that we didn’t have people who were engaging with algorithms’ role was exacerbating this existing inequality. With any sort of social issue, an algorithm can make things a lot worse, or it can help you understand what’s going on better and try to move things in a positive direction.”
Announcements

• APT-8 due Tuesday, April 20, Today!
• Assign 6 Recommender, due Thurs 4/22
  • One grace day, NO LATE DAYS!
  • MUST TURN in BY FRIDAY 4/23
• Assign 7 Create due, Friday, April 23!
  • Grace period is through Monday, April 26
  • No Late days!

• Lab 12 Friday, prelab available later today!

• Exam 3 back soon,...
Final Exam

• 3 hour exam – giving you 6 hours to take it

• 3 parts
  • PART A) on Sakai: (programming, like an APT Quiz)
    • 50 min – giving you 2 hours
    • Take any time April 27-29
  • PART B) – more programming, like Part B)
    • 50 min – giving you 2 hours
    • Take any time April 27-29
  • PART C) on GradeScope:
    • 80 min – giving you 2 hours
    • MUST BE taken on April 29.
Assignment 7: Create, Due Apr 23
Grace period til Apr 26, No late days!
Must be turned in by Apr 26
This assignment is required!

Pick one:
- Video: Green dance, advertisement for 101, song, other
- Poem or Multiple Haikus
- Story
- Comic
- One-pager
- Feedback

Let's see some examples
Video Simple Green Dance
Video: APT Success
PFTD

• How do Dictionaries work so fast!
  • Access an element in constant time

• Recursion
  • Solving a problem by solving smaller problems
How do Dictionaries work so fast?

- How are they implemented?
Simple Example
Want a mapping of Soc Sec Num to Names

• Duke's CS Student Union wants to be able to quickly find out info about its members. Also add, delete and update members. Doesn't need members sorted.
  267-89-5431   John Smith
  703-25-6141   Ademola Olayinka
  319-86-2115   Betty Harris
  476-82-5120   Rose Black

• Dictionary d – SSN to names
  • d[‘267-89-5431’] = ‘John Smith’
  • How does it find ‘John Smith’ so fast?
Dictionary \( d(\text{SSN}) = (\text{SSN}, \text{name}) \)

- We actually would map the SSN to the tuple of (SSN, name).
- That is a lot to display on a slide, so we will just show SSN to name.
- But remember name is really a tuple of (SSN, name)
Simple Example
Let’s look under the hood.
How are dictionaries implemented?

• Dictionaries implemented with a list, in a clever way
• How do we put something into the list fast?
• How do we find it in the list quickly?
  • \( d[‘267-89-5431’] = ‘John Smith’ \)
• List size is 11 – from 0 to 10
• \( d[‘267-89-5431’] \) calculates index location in list of where to put this tuple (SSN,name)
• Use a function to calculate where to store ‘John Smith’
  • \( H(ssn) = (\text{last 2 digits of } ssn) \mod 11 \)
  • Called a Hash function
Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  10
Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

\[
\begin{align*}
H(267-89-5431) &= 31 \mod 11 = 9 \\
\text{John Smith} \\
H(703-25-6141) &= 41 \mod 11 = 8 \\
\text{Ademola Olayinka} \\
H(319-86-2115) &= 15 \mod 11 = 4 \\
\text{Betty Harris} \\
H(476-82-5120) &= 20 \mod 11 = 9 \\
\text{Rose Black}
\end{align*}
\]

Collision!
Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

H(267-89-5431) = 31 %11 = 9
  John Smith

H(703-25-6141) = 41%11 =  8
  Ademola Olayinka

H(319-86-2115 )= 15 %11 =  4
  Betty Harris

H(476-82-5120) = 20%11 = 9
  Rose Black

Must resolve collisions
When does this work well?

• When there are few collisions

• You have to deal with collisions

• Use a list large enough to spread out your data
Another way: Use a list of lists

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

H(267-89-5431) = 31 % 11 = 9
  John Smith
H(703-25-6141) = 41%11 = 8
  Ademola Olayinka
H(319-86-2115) = 15 % 11 = 4
  Betty Harris
H(476-82-5120) = 20%11 = 9
  Rose Black

4/20/21
Compisci 101, Spring 2021 20
Another way: Use a list of lists

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

H(267-89-5431) = 31 % 11 = 9
  John Smith
H(703-25-6141) = 41%11 = 8
  Ademola Olayinka
H(319-86-2115) = 15 % 11 = 4
  Betty Harris
H(476-82-5120) = 20%11 = 9
  Rose Black

Collisions added to list, 2 in list 9
Recursion

- Solving a problem by solving similar but smaller problems
Recursion
Solving a problem by solving similar but smaller problems

**Question** - How many rows are there in this classroom?

**Similar but smaller question** - How many rows are there until your row?

<table>
<thead>
<tr>
<th>Row Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

Return value = 1

Return Value = 3+1 = 4

Return Value = 2+1 = 3

Return Value = 1+1 = 2

Return value = 1

So I am in Row#1
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

Level 0

Folder 1

Level 1

Folder 2

Folder 3

Level 2

Folder 4

Folder 5

Level 3

Folder 6

Level 4

Base Case

4/20/21

Compsci 101, Spring 2021
Recursion to print ALL files in a folder

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

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

Is that a directory?

If not a directory, it will be a file
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:
#bigs = bigfiles("/Users/Susan/Documents",10000)
# on Windows like this:
bigs = 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/101s21-0420-2
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\rodger\foo or /Users/rodger/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 …

\[
\begin{array}{c}
[ [ [a,b], [c,d], [a, [b,c],d] ] \\
(a *(b + c (d + e*f)) + (a* (b+d)))
\end{array}
\]
Recursion Summary

• Make Simpler or smaller calls
  • Call a clone of itself with different input
• 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
Mystery Recursion
bit.ly/101s21-0420-3
def Mystery(num):
    if num > 0:
        return 1 + Mystery(num//2)
    else:
        return 2 + num
Example

```python
def Mystery(num):
    if num > 0:
        return 1 + Mystery(num//2)
    else:
        return 2 + num
```

- Mystery(4) is \(1 + \text{Mystery}(2)\) = 1 + 4 = 5
- Mystery(2) is \(1 + \text{Mystery}(1)\) = 1 + 3 = 4
- Mystery(1) is \(1 + \text{Mystery}(0)\) = 1 + 2 = 3
- Mystery(0) is 2
Example

def Mystery(num):
    if num > 0:
        return 1 + Mystery(num//2)
    else:
        return 2 + num

• Mystery(18) is \( 1 + \text{Mystery}(9) \) = \( 1 + 6 = 7 \)
• Mystery(9) is \( 1 + \text{Mystery}(4) \) = \( 1 + 5 = 6 \)
• Mystery(4) is \( 1 + \text{Mystery}(2) \) = \( 1 + 4 = 5 \)
• Mystery(2) is \( 1 + \text{Mystery}(1) \) = \( 1 + 3 = 4 \)
• Mystery(1) is \( 1 + \text{Mystery}(0) \) = \( 1 + 2 = 3 \)
• Mystery(0) is \( 2 + 0 \)
Mystery in Python Tutor

Python 3.6
(known limitations)

```python
1 def Mystery(num):
2     if num > 0:
3         return 1 + Mystery(num//2)
4     else:
5         return 2 + num
7 7 if __name__ == '__main__':
8      print("Mystery(7) is", Mystery(7))
```

Edit this code

11/12/2020  Compsci 101, Fall 2020