CompSci 101
Recursion

Susan Rodger
December 6, 2022
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-7 due Thursday!

• Assign 6 Recommender, due tonight
  • One grace day, **NO LATE DAYS! NO Extensions!**
  • MUST TURN in BY tomorrow!!

• Assign 7 Create due, Friday, December 9!
  • Grace period with no penalty thru Tuesday, Dec 13
  • No Late turnins or extensions after Dec 13!

• Lab 11 Friday, do prelab!

• Final Exam – 7pm, Thursday, December 15
  • 3 hours, in person, covers topics through last day
Two extra credit opportunities!

• Fill out survey/consent for study (emailed last night)
  • 70% fill out by Friday – 1 extra point on Exam 3

• CompSci 101 course evaluation!
  • 70% fill out by Friday – 1 extra point on final exam
  • 80% fill out by Friday – a second extra point on final exam
More samples for Assignment 7
Compsci 101 Video
Video: APT Success
PFTD

• **Recursion**
  • Technique for solving a problem by solving smaller problems
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?

```
S
S
S
S
S
```

- I don’t know, let me ask
- I don’t know, let me ask
- I don’t know, let me ask
- I don’t know, let me ask
- I don’t have anyone to ask. So I am in Row#1

Row count = 4+1 = 5
Return Value = 3+1 = 4
Return Value = 2+1 = 3
Return Value = 1+1 = 2
Return value = 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  Level 1  Level 2  Level 3  Level 4

Folder 1
Folder 2  Folder 3
Folder 4  Folder 5
Folder 6

Base Case

Compsci 101, Fall 2022
Recursion (idea) 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
Recursion (idea) 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))
```

Name of function is `visit`

Call `visit` on a smaller problem

Is that a directory?

If not a directory, it will be a file
Recursion (idea) 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))
```

Calling the function we are defining, but on a SMALLER problem

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)
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/101f22-1206-1
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/files1.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!
Does the function call itself? No!

```python
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!

Function calls a smaller instance of itself, on smaller data.
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 ...

• **Is e in this list?**
  • How many lists do you have to look in?

\[
\text{[ } \text{ [ [a,b], [c,d] ]}, \text{ [a, [b,c],d]} \text{ ]}
\]
Structure matches Code

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

• **Is e in this list?**
  • How many lists do you have to look in?

\[
[ \left[ \left[ a, b \right], \left[ c, d \right] \right], \left[ a, \left[ b, c \right], d \right] ]
\]
Structure matches Code

• **Structure of expressions**
  • Can also lead to processing an expressions which requires processing an expression...

• **How do you evaluate expression?**

\[(a \times (b + c + (d + e\times f)) + (a\times (b+d))))\]
Structure matches Code

• Structure of list of lists
  • Can also lead to processing an expressions which requires processing an expression...

• How do you evaluate expression?
  \((a \times (b + c + (d + e \times f)) + (a \times (b+d)))\)
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

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

```python
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 $2 + 0$
  \[= 2 + 0 = 2\]
Mystery in Python Tutor

Python 3.6

(known limitations)

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

Print output (drag lower right corner to resize)

Frames

Objects

Global frame

Mystery

function Mystery(num)

Mystery

num | 7

Mystery

num | 3

Mystery

num | 1

Mystery

num | 0

Return value | 2

Edit this code

just executed

to execute

Step 16 of 19

qualification (NEW!)
Something Recursion
bit.ly/101f22-1206-3
**Something Recursion**

What is Something([3,5,1])?

```python
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:  # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])
```
Something Recursion

```python
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:  # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])
```

- Something([3, 5, 1]) is 0
- Something([5, 1]) = 0
- Something([1]) = 0
- Something([ ]) = 0

Something([3, 5, 1]) is 0
What is \( \text{Something}([5,4,2,3]) \) ?

```python
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:  # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])
```
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:
        # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])

Something([5,4,2,3]) is 6

• Something([5,4,2,3]) is
• Something([4,2,3]) is
• Something([2,3]) is
• Something([3]) is
• Something([ ]) is

Something([4,2,3]) = 6
4 + Something([2,3]) = 6
2 + Something([3]) = 2
Something([ ]) = 0

0
Recursion in Pictures

Enjoy a cookie!
You get one half of python logo
blue or yellow python