CompSci 101
Recursion

Susan Rodger
April 18, 2023
W is for ...

- World Wide Web
  - Where http meets tcp/ip?
- WiFi
  - We need and use this every day
- Windows
  - From OS to ...
“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 in one week!

• Assign 6 Recommender, due Thursday
  • Assign 6 Sakai quiz due tonight!

• Assign 7 Create due, Wednesday, April 26!
  • No penalty thru Sunday, April 30

• Lab 11 Friday, do prelab!

• Final Exam – 9am, Thursday, May 4
  • 3 hours, in person, covers topics through last day
Extra credit opportunity!

• Fill out survey under Exam 3 Bonus in Sakai tests and quizzes
  • If 65% fill out 1 extra credit point for Exam 3
  • If 75% fill out 1 additional extra credit point Exam 3

• Right now at 44%
More samples for Assignment 7
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?

<table>
<thead>
<tr>
<th>Row</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1+1 = 2</td>
</tr>
<tr>
<td>3</td>
<td>2+1 = 3</td>
</tr>
<tr>
<td>4</td>
<td>3+1 = 4</td>
</tr>
<tr>
<td>5</td>
<td>4+1 = 5</td>
</tr>
</tbody>
</table>

Row count = 4+1 = 5

Return Value = 3+1 = 4

Return Value = 2+1 = 3

Return Value = 1+1 = 2

Return value = 1

I don’t have anyone to ask. 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?
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))
```
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
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:
#bigs = bigfiles("/Users/Susan/Documents",10000)
# on Windows like this:
bigs = bigfiles("C:\Users\Susan\Documents",10000)
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:
#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)
- ...

4/18/23
Finding Large Files questions
bit.ly/101s23-0418-1
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/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!
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!
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?

[ [ [a,b], [c,d] ], [a, [b,c],d] ]
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?

\[
\begin{bmatrix}
  \begin{bmatrix} [a, b], [c, d] \end{bmatrix}, [a, [b, c], d]\end{bmatrix}
\]

This structure fits with recursion
Structure matches Code

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

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

\[(a \times (b + c + (d + e \times f)) \times (a \times (b + d)))\]
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** $2 + 0$
```python
def Mystery(num):
    if num > 0:
        return 1 + Mystery(num//2)
    else:
        return 2 + num

if __name__ == '__main__':
    print("Mystery(7) is", Mystery(7))
```

**Print output**

Frames

Objects

Global frame

Mystery

function

Mystery(num)

Mystery

num 7

Mystery

num 3

Mystery

num 1

Mystery

num 0

Return value

2
Something Recursion
bit.ly/101s23-0418-3
What is Something([3,5,1])?

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:])
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
• Something([5,1]) is
• Something([1]) is
• Something([]) is

Something([3,5,1]) is 0
Something Recursion

What is Something([5,4,2,3])?

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:])

4/18/23
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]) = 6
- Something([4,2,3]) is
- Something([2,3]) = 6
- Something([2,3]) is
- Something([2,3]) = 6
- Something([2,3]) is
- Something([3]) = 2
- Something([3]) is
- Something([3]) = 2
- Something([3]) is
- Something([3]) = 0
- Something([3]) is
- Something([ ]) = 0
- Something([ ]) is
- Something([ ]) = 0
- Something([ ]) is
- Something([ ]) = 0
Recursion in Pictures

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