CompSci 101
Introduction to Computer Science

Feb 21, 2017
Prof. Rodger
Announcements

• Reading and RQ due next time
• APT 4 out today, due Feb 28
• Do not discuss exam1 with anyone until it is handed back, likely Thursday
• Lab this week – undetermined repetition

• Today:
  – Loops – While, While True
  – Problem Solving
Developing an Algorithm

- [http://www.youtube.com/watch?v=AEBbsZK39es](http://www.youtube.com/watch?v=AEBbsZK39es)

$193, $540, $820, $700, $749. Are these reasonable? Why?
I'm thinking of a number …

• You guess. I'll tell you high, low, or correct
  – Goal: guess quickly, minimal number of guesses
  – Number between 1 and 100…
  – Number between 1 and 1000…

• Can you describe an algorithm, instructions, that would allow someone to use your instructions to play this game correctly. Start with 1 and 100, but ideally your instructions work with 1 and N

bit.ly/101s17-0221-1

compsci 101, spring 2017
Analyzing the *binary search* algorithm

• Is the algorithm correct?
  – Try it, again, and again and …
  – Reason about it: logically, informally, …

• How efficient is the algorithm?
  – How many guesses will it take (roughly, exactly)
  – Should we care about efficiency?

• When do we really care about efficiency?
  – Examples?
Find Narten
Find Narten
Looking for a Needle in a Haystack

• If a computer can examine 10 million names/numbers a second, suppose the list isn't sorted, or I say "yes/no", not "high/low"
  – How long to search a list of 10 million?
  – How long to search a list of a billion?
  – 14 billion pixels in a 2 hour blu-ray movie

• What about using binary search? How many guesses for 1000, 10^6, 10^9, 10^{12}
  – One of the things to remember: \(2^{10} = 1024\)
Review - Searching for words

• If we had a million words in alphabetical order, how many would we need to look at worst case to find a word?
Review - Searching for words

- If we had a million words in alphabetical order, how many would we need to look at worst case to find a word?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000,000</td>
<td>976.56</td>
</tr>
<tr>
<td>500,000</td>
<td>488</td>
</tr>
<tr>
<td>250,000</td>
<td>244</td>
</tr>
<tr>
<td>125,000</td>
<td>122</td>
</tr>
<tr>
<td>62,500</td>
<td>61</td>
</tr>
<tr>
<td>31,250</td>
<td>30</td>
</tr>
<tr>
<td>15,625</td>
<td>15</td>
</tr>
<tr>
<td>7812.5</td>
<td>7.5</td>
</tr>
<tr>
<td>3906</td>
<td>3.75</td>
</tr>
<tr>
<td>1953</td>
<td>1.875</td>
</tr>
</tbody>
</table>

- 20 words!

If you are clever, cut the number of numbers to look at in half, over and over again.
Prime Numbers

• An integer > 1 is prime if it has no positive divisors other than 1 and itself.

• 12 is not prime!
  – 12 is divisible by 2, 3, 4, 6
  – $3 \times 4 = 12$, $2 \times 6 = 12$

• Prime numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23

• Is 8315411 prime?
Is number a Prime number?
Bit.ly/101s17-0221-2

def isPrime(number):
    if number < 2:    # must be greater than 1
        return False
    if number < 4:    # 2 and 3 are prime
        return True
    for n in range(4,number):
        if number/n * n == number:
            return False
    return True
APT PrimeTime

APT: PrimeTime

Problem Statement

A prime number is a number greater than 1 divisible only by 1 and itself. For example, 2, 3, 5, 7, 11, 13, 17 are primes, but 15 is not since it's divisible by 3 and 5.

Write function pcount that returns the number of prime numbers between low and high, inclusive.

Specification

```
filename: PrimeTime.py

def pcount(low, high):
    """
    return int, the number of prime numbers between int values low and high, inclusive
    """

    # you write code here
```
Write Helper functions to help solve problems!!!!

• APT PrimeTime
  – Use isPrime as a helper function

• Assignment 4 helper functions
  – isVowel(letter) – return true if letter is a vowel
  – NoVowels(word) – return True if no vowels in word
  – Automatic Decrypt, what helper function?
Write Helper functions to help solve problems!!!!

- APT PrimeTime
  - Use isPrime as a helper function

- Assignment 4 helper functions
  - isVowel(letter) – return true if letter is a vowel
  - NoVowels(word) – return True if no vowels in word
  - Automatic Decrypt, what helper function?
    - countWords(wordlist, shift, phrase)
    - Decrypt with shift, then count how many words in phrase are in wordlist
Undetermined Repetition

• Game of chess, when does it end?

• What is the 100\textsuperscript{th} prime number?

• Guessing a number from 1 to 100?
While loops

• Repetition when you stop a loop based on a condition

• while CONDITION:
  BODY

  – As long as condition is true, keep executing loop.
  – Must have an update in the body to get closer to condition being false
Example for while

- Playing chess

while (game not over)
   make a move in the game
   (game must get closer to ending)
Example2 for while

- What is the 100\textsuperscript{th} prime number?

\begin{verbatim}
number = 2
while (not 100\textsuperscript{th} prime)
    is number prime?
    update count
    generate next number to check
\end{verbatim}

(program must get closer to ending)
Example3 - Factorial

- $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$
- $3! = 3 \times 2 \times 1 = 6$
Example with while loop

def factorial(num):
    result = 1
    while num > 0:
        result = result * num
        num = num - 1
    return result

for n in range(8):
    print n, factorial(n)
def mystery(string, letter):
    pos = 0
    count = 0
    result = ''
    while count < 4 and pos < len(string):
        if string[pos] == letter:
            result += string[pos] + string[pos]
            count += 1
        else:
            result += string[pos]
        pos += 1
    return result

print mystery("September December", "e")
The 21 Most Important Googlers You've Never Heard Of

Georges Harik and Noam Shazeer created the underlying data that led to AdSense

Harik and Shazeer spent years analyzing data on webpages, trying to understand clusters of words and how they worked together. The data they gather wound up being used by Google for its AdSense product, which analyzed webpages for words, and then stuck ads on them.
Looping with while
– not sure when to stop

• Playing chess
• Determining the 100th prime number

• Another way – while True
  – Must have ways to break out of infinite loop
  – Must have update – gets closer to ending
while condition vs while True

while condition:
    body
    continue

while True:
    body
    if condition:
        break
        continue

While condition is true - must update
- must get closer to making condition false
- use break to exit
Format of While True

initialize
while True:
    if something:
        break
    if something2:
        update
    update
Continue or return
Revisit Factorial with while True

def factorial(num):
    result = 1
    while True:
        if num == 0:
            break
        result = result * num
        num = num - 1
    return result
def mystery2(string, letter):
    pos = 0
    count = 0
    result = ''
    while True:
        # missing code to break out of while
        if string[pos] == letter:
            result += string[pos] + string[pos]
            count += 1
        else:
            result += string[pos]
        pos += 1
    return result
Problem: Find the location of first adjacent duplicate word

bit.ly/101s17-0221-5

• “This is a story about a a girl with a red hood…”

• Return six as the location of the second word “a”