CompSci 101
Introduction to Computer Science

Oct 6, 2016
Prof. Rodger
Announcements

• Reading and RQ due next time
• APT 4 out today, due Oct 18
• Do not discuss exam1 with anyone until it is handed back, likely after fall break
• No Lab this week or next week

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

- 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/101f16-1006-1
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

FOUND!
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?

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

<table>
<thead>
<tr>
<th></th>
<th>1,000,000</th>
<th>976.56</th>
</tr>
</thead>
<tbody>
<tr>
<td>500,000</td>
<td>488</td>
<td></td>
</tr>
<tr>
<td>250,000</td>
<td>244</td>
<td></td>
</tr>
<tr>
<td>125,000</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>62,500</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>31,250</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>15,620</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>7,812.5</td>
<td>7.5</td>
<td></td>
</tr>
<tr>
<td>3,906</td>
<td>3.75</td>
<td></td>
</tr>
<tr>
<td>1,953</td>
<td>1.875</td>
<td></td>
</tr>
</tbody>
</table>

- 20 words!

If you are clever, cut the number of numbers to look at in half, over and over again.
Is number a Prime number?
Bit.ly/101f16-1006-2

def isPrime(number):
    if number<4:
        return True
    for n in range(4,number):
        if number/n * n == number:
            return False
    return True
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
Examples for while

- Playing chess

while (game not over)

play game

(game must get closer to ending)

- Finding the 100\textsuperscript{th} prime
Mystery While example

bit.ly/101f16-1006-3

def mystery(strng):
    count = 0
    result = ""
    while count < 5:
        result += strng[count] + strng[count]
        count += 1
    result += strng[count:]
    return result

print mystery("September")
Problem: Given a number, want the largest list of unique digits from 1 to x whose sum is less than or equal to the number

• Given 5
  Answer is $1 + 2$, list [1,2]

• Given 6
  Answer is $1+2+3$, list [1,2,3]
def addDigitsTilSum2(total):
    sum = 0
    digs = []
    for n in range(1,10):
        sum += n
        if sum > total:
            break
    digs.append(n)
return digs
Looping with while
– not sure when to stop

• Playing chess
• Determining the 100\textsuperscript{th} 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
While True

initialize

while True:
    if something:
        break
    if something2:
        update
            update

Continue or return
def addDigitsTilSum(total):
    sum = 0
    num = 1
    digs = []
    while(True):
        sum += num
        if sum > total:
            break
        digs.append(num)
        num += 1
    return digs
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.