Announcements

- Assign 6 due extended one day
  - Assign 7 out today, due Nov 29
- APT 9 due Tuesday (No extensions)
- Next week – No lab, Exam Thursday
- Practice exams – work on for next class

- Today:
  - Why are dictionaries so fast?
  - More problem solving with dictionaries

Be in the know….
ACM, compsci mailing lists

- Association of Computing Machinery (ACM)
  - Professional organization for computer science
  - Duke Student ACM Chapter – join for free
- Join duke email lists to find out info on jobs, events for compsci students
  - lists.duke.edu – join lists:
    - compsci – info from compsci dept
    - dukeacm – info from student chapter

Dictionary Song problem
bit.ly/101f16-1110-1

Problem: (word,count of words)
- Updating (key,value) pairs in structures
- Three different ways:
  1. Search through unordered list
  2. Search through ordered list
  3. Use dictionary
- Why is searching through ordered list fast?
  - Guess a number from 1 to 1000, first guess?
  - What is $2^{10}$? Why is this relevant? $2^{20}$?
  - Dictionary is faster! But not ordered

Linear search through list o' lists
- Maintain list of [string,count] pairs
  - List of lists, why can't we have list of tuples?
  
  ```
  ['dog', 2], ['cat', 1], ['bug', 4], ['ant', 5]
  ```
  - If we read string 'cat', search and update
  
  ```
  ['dog', 2], ['cat', 2], ['bug', 4], ['ant', 5]
  ```
  - If we read string 'frog', search and update
  
  ```
  ['dog', 2], ['cat', 2], ['bug', 4], ['ant', 5], ['frog', 1]
  ```

See DictionaryTimings.py
```python
def linear(words):
    data = []
    for w in words:
        found = False
        for elt in data:
            if elt[0] == w:
                elt[1] += 1
                found = True
                break
        if not found:
            data.append([w,1])
    return data
```
Binary Search

Find Narten

How many times
divide in half?

\[ \log_2(N) \text{ for } N \text{ element list} \]

Binary search through list o' lists

- Maintain list of [string,count] pairs in order

\[
[ ['ant', 4], ['frog', 2] ]
\]
- If we read string 'cat', search and update

\[
[ ['ant', 4], ['cat', 1], ['frog', 2] ]
\]
- If we read string 'dog' twice, search and update

\[
[ ['ant', 4], ['cat', 1], ['dog', 1], ['frog', 2] ]
\]
\[
[ ['ant', 4], ['cat', 1], ['dog', 2], ['frog', 2] ]
\]

See DictionaryTimings.py

bit.ly/101f16-1110-3

def binary(words):
    data = []
    for w in words:
        elt = [w,1]
        index = bisect.bisect_left(data, elt)
        if index == len(data):
            data.append(elt)
        elif data[index][0] != w:
            data.insert(index,elt)
        else:
            data[index][1] += 1
    return data

Search via Dictionary

- In linear search we looked through all pairs
- In binary search we looked at log pairs
  - But have to shift lots if new element!!
- In dictionary search we look at one pair
  - Compare: one billion, 30, 1, for example
  - Note that \( 2^{10} = 1024, 2^{20} = \text{million}, 2^{30}=\text{billion} \)
- Dictionary converts key to number, finds it
  - Need far more locations than keys
  - Lots of details to get good performance
```python
def dictionary(words):
    d = {}
    for w in words:
        if w not in d:
            d[w] = 1
        else:
            d[w] += 1
    return [[w, d[w]] for w in d]
```

Running times @ $10^9$ instructions/sec

<table>
<thead>
<tr>
<th>$N$</th>
<th>$O(\log N)$</th>
<th>$O(N)$</th>
<th>$O(N \log N)$</th>
<th>$O(N^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^2$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00001</td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.0</td>
<td>0.0000001</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>$10^6$</td>
<td>0.0</td>
<td>0.001</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>$10^9$</td>
<td>0.0</td>
<td>1.0</td>
<td>29.9</td>
<td></td>
</tr>
<tr>
<td>$10^{12}$</td>
<td>9.9 secs</td>
<td>16.7 min</td>
<td>11.07 hr</td>
<td></td>
</tr>
</tbody>
</table>

This is a real focus in Compsci 201

Linear is $N^2$, binary is $N \log N$, dictionary $N$

What's the best and worst case?

Bit.ly/101f16-1110-4

- If every word is the same ....
  - Does linear differ from dictionary? Why?
- If every word is different in alphabetical ...
  - Does binary differ from linear? Why?
- When would dictionary be bad?

Next Assignment –
Clever, Snarky, Evil, Frustrating Hangman

- Computer changes secret word every time player guesses to make it "hard" to guess
  - Must be consistent with all previous guesses
  - Idea: the more words there are, harder it is
    - Not always true!
- Example of greedy algorithm
  - Locally optimal decision leads to best solution
  - More words to choose from means more likely to be hung
Canonical Greedy Algorithm

• How do you give change with fewest number of coins?
  – Pay $1.00 for something that costs $0.43
  – Pick the largest coin you need, repeat

Greedy not always optimal

• What if you have no nickels?
  – Give $0.31 in change
  – Algorithms exist for this problem too, not greedy!

Clever Hangman

• When you guess a letter, you're really guessing a category (secret word "salty")
  _ _ _ _ _ and user guesses 'a'
  – "gates", "cakes", "false" are all the same
  – "flats", "aorta", "straw", "spoon" are all different

• How can we help ensure player always has many words to distinguish between?

Debugging Output

number of misses left: 8
secret so far: _ _ _ _ _ _ _ (word is catalyst)
# possible words: 7070
guess a letter: a
  a__a__a 1
  ... 
  _a_____ 587
  __aa___ 1
  ... 
  __a____ 498
  _______ 3475
  ___a____ 406
  ... 
  _____a__ 396
  # keys = 48

number of misses left: 7
letters not yet guessed: bcdefghijklmnopqrstuvwxyz...
(word is designed)
# possible words: 3475
guess a letter:
Debugging Output and Game Play

- Sometimes we want to see debugging output, and sometimes we don't
  - While using Microsoft Word, don't want to see the programmer's debugging statements
  - Release code and development code

- You'll approximate release/development using a global variable DEBUG
  - Initialize to False, set to True when debugging
  - Ship with DEBUG = False

Look at howto and categorizing words

- Play a game with a list of possible words
  - Initially this is all words
  - List of possible words changes after each guess

- Given template "_ _ _ _", list of all words, and a letter, choose a secret word
  - Choose all equivalent secret words, not just one
  - Greedy algorithm, choose largest category

Computing the Categories

- Loop over every string in words, each of which is consistent with guess (template)
  - This is important, also letter cannot be in guess
  - Put letter in template according to word
    - _ _ _ a _ t might become _ _ _ a n t

- Build a dictionary of templates with that letter to all words that fit in that template.
- How to create key in dictionary?

Dictionary to help solve…

- Example: Four letter word, guess o

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;O _ O _&quot;</td>
<td>&quot;OBOE&quot;, &quot;ODOR&quot;</td>
</tr>
<tr>
<td>&quot;_ O O _&quot;</td>
<td>&quot;NOON&quot;, &quot;ROOM&quot;, &quot;HOOP&quot;</td>
</tr>
<tr>
<td>&quot; _ O _ O&quot;</td>
<td>&quot;SOLO&quot;, &quot;GOTO&quot;</td>
</tr>
<tr>
<td>&quot;_ _ _ O&quot;</td>
<td>&quot;TRIO&quot;</td>
</tr>
<tr>
<td>&quot;O _ _ _ &quot;</td>
<td>&quot;OATH&quot;, &quot;OXEN&quot;</td>
</tr>
<tr>
<td>&quot;_ _ _ _&quot;</td>
<td>&quot;PICK&quot;, &quot;FRAT&quot;</td>
</tr>
</tbody>
</table>

- Key is string, value is list of strings that fit
Keys can’t be lists

• [“O”,”_”,”O”,”_”] need to convert to a string to be the key representing this list:
  “O_O_”