Announcements

- Assign 6 due today
  - Assign 7 out later today, due in two weeks
- APT 9 due Tuesday
- Next week – No lab, focus on exam
- 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

Events for Women in CS

- Pearl hacks
- UNC April 2-3
- Diamond Hacks
- NCSU Apr 16-17
Events for Women in CS (cont)

- Grace Hopper Celebration of Women in Computing
  - *Apply now for full scholarships* – need one letter
  - Deadline is Monday, April 4
  - Conference is October 19-21

ACM Richard Tapia Celebration of Diversity In Computing

Sept. 14-17, 2016

Dictionary Song problem

```
songs = [
    "Hey Jude:Let it be:Day Tripper",
    "Let it be:Drive my car:Hey Jude",
    "I want to hold your hand:Day Tripper:Help!",
    "Born to run:Thunder road:She's the one",
    "Hungry heart:The river:Born to run",
    "The river:Thunder road:Drive my car",
    "Angie:Start me up:Ruby Tuesday",
    "Born to run:Angie:Drive my car"
]
```

APT Emails Course

```
You are given a list of strings of course information, where each string is in the format course:person:email. Your task is to determine the course with the most people and to return the emails of those people in the largest course. The emails should be returned as a string with the emails in alphabetical order. If there is more than one largest course, return the emails of such course that comes first in alphabetical order.

```
["CompSci 100:Fred Jack Smith:fjs@duke.edu",
 "History 117:Fred Jack Smith:fjs@duke.edu",
 "CompSci 102:Arielle Marie Johnson:amj@duke.edu",
 "CompSci 100:Arielle Marie Johnson:amj@duke.edu",
 "CompSci 006:Bertha White:bw@duke.edu",
 "Econ 051:Bertha White:bw@duke.edu",
 "English 112:Harry Potter:hp@duke.edu",
 "CompSci 100:Harry Potter:hp@duke.edu"]
```

Returns "amj@duke.edu fjs@duke.edu hp@duke.edu"
DictionaryTimings.py
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
```

N new words?

1. Anderson
2. Applegate
3. Bethune
4. Brooks
5. Carter
6. Douglas
7. Edwards
8. Franklin
9. Griffin
10. Holhouser
11. Jefferson
12. Klatchy
13. Morgan
14. Munson
15. Narten
16. Oliver
17. Parker
18. Rivers
19. Roberts
20. Stevenson
21. Thomas
22. Wilson
23. Woodrow
24. Yarbrow

Binary Search

Find Narten

```
1. Anderson
2. Applegate
3. Bethune
4. Brooks
5. Carter
6. Douglas
7. Edwards
8. Franklin
9. Griffin
10. Holhouser
11. Jefferson
12. Klatchy
13. Morgan
14. Munson
15. Narten
16. Oliver
17. Parker
18. Rivers
19. Roberts
20. Stevenson
21. Thomas
22. Wilson
23. Woodrow
24. Yarbrow
```

FOUND!

How many times divide in half?

$\log_2(N)$ for $N$ element list

compsci 101 spring 2016
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

```
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
```
**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>0.00000001</td>
<td>0.000001</td>
<td>0.001</td>
</tr>
<tr>
<td>10^4</td>
<td>0.0</td>
<td>0.001</td>
<td>0.02</td>
<td>16.7 min</td>
</tr>
<tr>
<td>10^9</td>
<td>0.0</td>
<td>1.0</td>
<td>29.9</td>
<td>31.7 years</td>
</tr>
<tr>
<td>10^12</td>
<td>9.9 secs</td>
<td>16.7 min</td>
<td>11.07 hr</td>
<td>31.7 million years</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/101sp16-0331-4

- 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!

![Nickels](image)

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 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

```plaintext
number of misses left: 8
secret so far: _ _ _ _ _ _ _ _
(word is catalyst)
# possible words: 7070
guess a letter: a
  a_a__a 1
  ...  
  _a_____ 587
  ___a__a 1
  ...  
  __a___ 498
  ____a__ 3475
  ___a__ 406
  ...  
  ___a_ 396
# keys = 48
```
Look at how to 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_ _&quot;</td>
<td>&quot;GBOE&quot;, &quot;ODOR&quot;</td>
</tr>
<tr>
<td>&quot;<em>O</em> _&quot;</td>
<td>&quot;NOON&quot;, &quot;ROOM&quot;, &quot;HOOP&quot;</td>
</tr>
<tr>
<td>&quot; _ _ O&quot;</td>
<td>&quot;SOLO&quot; &quot;GOTO&quot;</td>
</tr>
<tr>
<td>&quot; _ _ _&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_"