Plan for eleven-four

- Thinking about APTs and test problems
  - How do you choose: list, string, set, dictionary
  - Experience? How do you get that?
  - Most APTs and test problems share structure:
    - There's a loop, there's a selection/decision, update

- You can often do this with a list comprehension, but you don't have to!
  - Write code you can understand, but you must be able to read code with list comprehensions and with dictionaries
SortedFreqs


- **What do you return? How many elements does it contain? Can you categorize them?**
  - Read problem, understand what to return
  - Then think about how to calculate/create values

- **Is efficiency an issue with APTs?**
  - Computers do millions of operations a second
  - Your time is important!
  - Always possible to get time-limit exceeded 😊
Customer Statistics


- What's returned? How many elements does it contain? Can you categorize them?
  - Read problem, understand what to return
  - Then think about how to calculate/create values

- How can you find names that occur more than once? Can you filter names/elements?
  - Filtering is a great use of list comprehensions!
  - Creating return values in correct order, issues?
Questions

Shafi Goldwasser

- 2012 Turing Award Winner
- RCS professor of computer science at MIT
  - Twice Godel Prize winner
  - Grace Murray Hopper Award
  - National Academy
  - Co-inventor of zero-knowledge proof protocols

How do you convince someone that you know [a secret] without revealing the knowledge?

- Honesty and Privacy

Work on what you like, what feels right, I know of no other way to end up doing creative work
DictionaryTimings.py

- Updating \((key, value)\) pairs in structures
  - Search through unordered list
  - Search through ordered list
  - 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]]
```
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

- Before the first guess, there are 1024 numbers

How many times can we divide list in half?

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

What must be true to use binary search?

How is this done in Python?
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
  - 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
See DictionaryTimings.py

- Finding value associated with key w:
  - Takes time independent of number of keys!

```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>0.000001</td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.0</td>
<td>0.0000001</td>
<td>0.00001</td>
<td>0.001</td>
</tr>
<tr>
<td>$10^6$</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?

- If every word is the same ....
  - Does linear differ from dictionary? Why?
- Every word is different in alphabetical order
  - Does binary differ from linear? Why?
- When would dictionary be bad?
  - In practice, never, in theory, kind of the same
Practice Test Question


● Read, think, read, think, plan, think, write
  ➢ If you're not sure, come back to question
  ➢ We won't ask you to write too much
  ➢ It's ok to write a lot if you can't write a little