Objectives

• Search strategies

Lab 10 Notes

• Advice:
  ➢ Read through entire lab to see where you are going
• A Person’s network is different from the SocialNetwork
  ➢ Person’s network is just a string
  ➢ May do more with that later…
• Can have people in different networks in the same SocialNetwork

Final Exam Details

• Discuss content later
  ➢ Focus since last exam
• All CS exams are taken in Parmly 405 (our lab)
• At your specified time, someone brings the tests to Parmly 405
• You have 3 hours to take the exam
• Can change exam time by using sheet outside of department office (Parmly 407)

Course Evaluations

• Next Wednesday
• General questions about the course
• Specific questions
  ➢ Feedback on improving the broader issues component of the course

Need 4 Volunteers

• No one will get hurt …

Find the Card in Your Deck

• Reminder to me: take out the jokers
• Challenge: who can find the card first
  ➢ (Most efficient algorithm)
• Need rest of class to keep searchers honest and help me determine who found the card first
The Race is On!
- 3 of Hearts
- 2 of Diamonds
- 4 of Clubs
- Queen of Spades
- King of Queens

Searching for a Playing Card
- Given a deck of cards and a card to find, describe the algorithm for how you would find that card.
  - Present several algorithms (naïve ones too!)
  - Discuss the strengths and weaknesses of each

Search Using \texttt{in} Review
- Iterates through a list, checking if the element is found
- Known as \textit{linear search}
- Implementation:

```python
def inSearch(searchlist, key):
    for elem in searchlist:
        if elem == key:
            return True
    return False
```

Search Using \texttt{in} Review
- \textbf{Overview}: Iterates through a list, checking if the element is found
- Known as \textit{linear search}
- \textbf{Benefits}:
  - Works on any list
- \textbf{Drawbacks}:
  - Does not tell us where in the list it is
    - What if wanted to do something to that element?
    - Could implement our own version that returns the position
  - Slow -- needs to check each element of list if the element is not in the list

High-Low Game
- I'm thinking of a number between 1-100
- You want to guess the number as quickly as possible (in fewest guesses)
- For every number you guess, I'll tell you if you got it right. If you didn't, I'll tell you whether you're too high or too low

Reminder: write down guesses

High-Low Game
- I'm thinking of a number between 1-100
- You want to guess the number as quickly as possible (in fewest guesses)
- For every number you guess, I'll tell you if you got it right. If you didn't, I'll tell you whether you're too high or too low

What is your best guessing strategy?
Strategy: Eliminate Half the Possibilities

• Repeat until find value (or looked through all values)
  ➢ Guess middle value of possibilities
  ➢ If match, found!
  ➢ Otherwise, find out too high or too low
  ➢ Modify your possibilities
    • Eliminate the possibilities from your number and higher/lower, as appropriate

• Known as Binary Search

Searching...

<table>
<thead>
<tr>
<th>-3</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Searching for 8

<table>
<thead>
<tr>
<th>-3</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

• Find the middle of the list
  ➢ Positions: 0 -- 7, so mid is ((7+0)/2) = 3
• Check if the key equals the value at mid (1)
  ➢ If so, report the location
• Check if the key is higher or lower than value at mid
  ➢ Search the appropriate half of the list

Searching for 8

• mid is 5 ((7+4)/2), list[5] is 7
  ➢ 8>7, so look in upper half
• mid is 6 ((7+6)/2), list[6] is 8
  ➢ 8==8, FOUND IT at position 6!

What if searched for 6 instead of 8?

<table>
<thead>
<tr>
<th>-3</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Searching for 6

<table>
<thead>
<tr>
<th>-3</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

• Will follow same program flow, but 6 is not in the list
• mid is 6, list[5] is 7
  ➢ 6 < 7, so will try to look in lower half of the list
• mid is 4, list[4] is 2
  ➢ 6>2, so will try to look in upper half of the list, but we've already determined it's not there.
  How do we know to stop looking?

Implementation Group Work

```python
def search(searchlist, key):
    """Pre: searchlist is a list of integers in sorted order. Returns the position of key (an integer) in the list of integers (searchlist) or -1 if not found""
    """
    # Trace through your program using examples
    ➢ Start simple (small lists)
    ➢ Do what the program says exactly, not what you think the program says
```
One Solution

```python
def search(searchlist, key):
    low = 0
    high = len(searchlist) - 1
    while low <= high:
        mid = (low + high) / 2
        if searchlist[mid] == key:
            return mid  # return True
        elif key > searchlist[mid]:
            low = mid + 1
        else:
            high = mid - 1
    return -1  # return False
```

Binary Search

- Example of a **Divide and Conquer** algorithm
  - Break into smaller pieces that you can solve
- Benefits:
  - Faster to find elements (especially with larger lists)
- Drawbacks:
  - Requires that data can be compared
    - `__cmp__` method implemented by the class
  - List must be sorted before searching
    - Takes time to search

Empirical Study of Search Techniques

- Goal: Determine which technique is better under various circumstances

- How long does it take to find various keys?
  - **Measure** by the number of comparisons
  - Vary the size of the list and the keys
  - What are good tests for the lists and the keys?

Empirical Study of Search Techniques

- Analyzing Results …
  - By how much did the number of comparisons for linear search vary?
  - By how much did the number of comparisons for binary search vary?
- What conclusions can you draw from these results?

For Friday

- Broader Issue
  - One of two social networking articles
- Lab 10