Outline Nov 12:

- Recap binary search and dictionary example
- Runtime of binary search
- Runtime worksheet (Handout 9)
- Sorting intro with LOL example
- Go over Quiz 3

Notes

- Lab 8 due Friday in-class (worksheet) and Saturday night
- Lab 9 due Monday after Thanksgiving
- Quiz 4 this Friday

Right now: sit somewhere new!
Tips for Lab 8

• Read the entire lab before starting!
• Use string formatting to right-align strings or numbers

```python
lst = ["anya","christina","clarissa","michelle","pravadh","rachel","rye","scout","tristan"]

for name in lst:
    print("%20s" % name)

    anya
    christina
    clarissa
    michelle
    pravadh
    rachel
    rye
    scout
    tristan
```
Binary Search & Dictionary Example
def binary_search(query, lst):
    low = 0
    high = len(lst) - 1
    count = 0

    # stop when low > high (strictly greater!)
    while low <= high:
        mid = int((low+high)/2)

        # case 1: our query was equal to the middle element: we're done!
        if query == lst[mid]:
            print("Number of comparisons:" , count)
            return mid

        # case 2: less than the middle element, must be in first half
        elif query < lst[mid]:
            high = mid - 1

        # case 3: greater than the middle element, must be in second half
        else:
            low = mid + 1

        count += 1  # one more comparison

    print("Number of comparisons:" , count)
    return -1
Reading the dictionary file

- All upper case words come before all lower case words

```python
def read_dictionary(filename):
    """Read a dictionary file and return a list of all the words."""
    word_file = open(filename, 'r')
    word_lst = []  # set up list accumulator

    # loop through each line (one word on each line)
    for line in word_file:
        word_lst.append(line.strip())

    word_file.close()
    return word_lst
```

```bash
>>> 'abba' < 'zip'
True
>>> 'abba' < 'Abba'
False
>>> 'Zip' < 'swarthmore'
True
```
Example of linear vs. binary

def main():
    # get a list of all words in the dictionary
dictionary = "/usr/share/dict/words"
word_lst = read_dictionary(dictionary)

    # obtain a smaller subset of the dictionary, print out first/last words
n = int(input("Enter the number of words to use: "))
subset = word_lst[:n]
print("First word", subset[0])
print("Last word", subset[-1])

    # ask the user for a word to search for
query = input("Enter a word to search for: ")

    # investigate the number of comparisons for linear and binary search
print("\nlinear search:")
index = linear_search(query, subset)
print(query, "is at index", index)

print("\nbinary search:")
index = binary_search(query, subset)
print(query, "is at index", index)
# Comparison examples

Enter the number of words to read: 1000000
start word: A
end word: études
Enter a word to search for: ninja

linear search:
Number of comparisons: 67475
ninja is at index 67475

binary search:
Number of comparisons: 16
ninja is at index 67475

Enter the number of words to read: 250000
start word: A
end word: études
Enter a word to search for: bat

linear search:
Number of comparisons: 24665
bat is at index 24665

binary search:
Number of comparisons: 14
bat is at index 24665

Enter the number of words to read: 250000
start word: A
end word: études
Enter a word to search for: Yeet

linear search:
Number of comparisons: 102304
Yeet is at index -1

binary search:
Number of comparisons: 17
Yeet is at index -1
Binary Search Runtime
Binary Search Runtime

\[ n = \# \text{ of elements} \]

\[ X = \# \text{ of comparisons.} \]

\[ \frac{n}{2} \geq 1 \]

\[ \log(n) = \log(2^x) \]

\[ x = \log_2(n) \]

For \( i \) in range(10):

- print(\( i \))

Constant runtime \( O(1) \)

Runtime \( = O(\log n) \)
### Linear vs. Binary: worst case

<table>
<thead>
<tr>
<th>n</th>
<th>Linear (# steps)</th>
<th>Binary (# steps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>10000</td>
<td>10000</td>
<td>14</td>
</tr>
<tr>
<td>100000</td>
<td>100000</td>
<td>17</td>
</tr>
<tr>
<td>1000000</td>
<td>1000000</td>
<td>20</td>
</tr>
</tbody>
</table>

![Graph](image1.png)

![Graph](image2.png)
Handout 9 (Runtime)
1. \( n \) steps
   \( O(n) \)
   linear
   100 steps
   \( O(1) \)
   constant

2. \( n + n \) steps
   \( 2n \)
   \( O(n) \)

3. \( n + n + n + n \ldots \)
   \( n \)
   \( O(n^2) \)

4. \( n + (n-1) + (n-2) + \ldots + 3 + 2 + 1 \)

5. \[ \sum_{i=0}^{n} i = \frac{n^2}{2} \]
   \( \# \text{ steps} \)

6. \( 10 + 10 + 10 + \ldots + 10 \)
   \( n \)
   \( 10n = \# \text{ steps} \)
   \( O(n) \)

7. \( O(\log n) \)

8. \( \# \text{ steps} = 4 \)
   \( O(1) \)

9. \( O(n \log(n)) \)
Quiz 3 common issues
1. Define the stack:
   - $s^a = \{7, 8, 9\}
   - $s^b = \{7, 8, 9\}$

2. Define the heap:
   - $\{7, 8, 9, 10\}$

3. Implement the `pick_card` function:
   ```python
   def pick_card(low, high):
     c = random.randint(low, high + 1)
   ```

4. Implement the main logic:
   ```python
   if $s^a[c] < \text{first_word}$:
     \text{first_word} = s^a[c]
   ```
   
   - $x = 3$
   - $y = x$
   - $y = y + 1$