CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2018

Swarthmore College

Outline Nov 12:

Right now: sit somewhere new!

- 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

Tips for Lab 8

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

```
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

Binary search: version with comparison counts

```
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

```
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
```

```
>>> '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 comparisions 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)
```

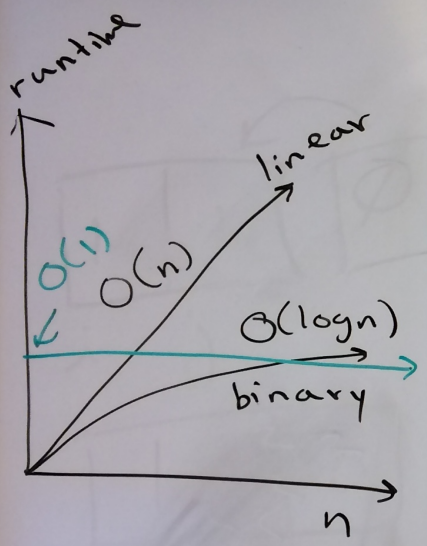
Comparison examples

```
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
```

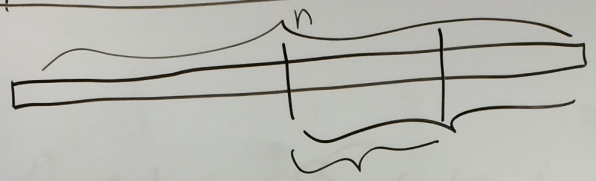
```
Enter the number of words to read: 250000
start word:  A
end word:  études
Enter a word to search for: Yeet

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

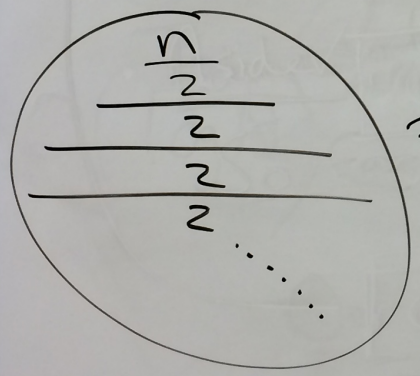


Binary Search Runtime

n = # of elements



X = # of comparisons.



≈ 1

$\frac{n}{2^x} = 1$

$\Rightarrow \log(n) = \log(2^x)$

$x = \log_2(n)$

runtime = $O(\log n)$

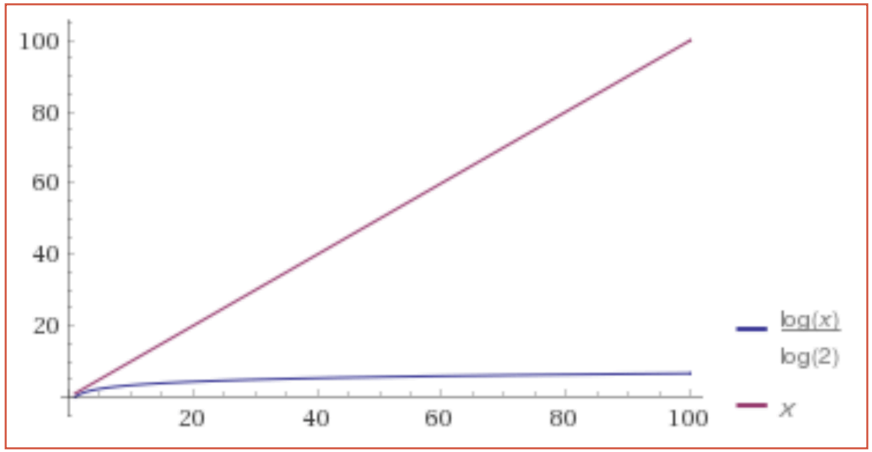
```
for i in range(10):
    print(i)
```

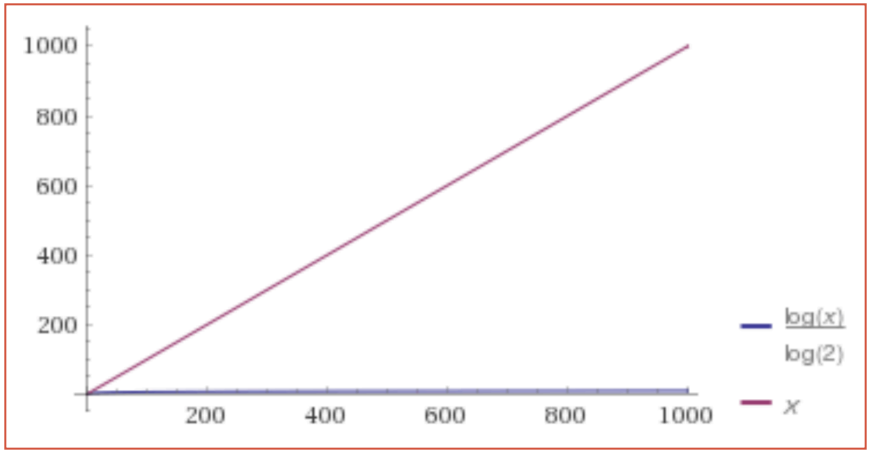
constant runtime

$O(1)$

Linear vs. Binary: worst case

n	Linear (# steps)	Binary (# steps)
100	100	7
1000	1000	10
10000	10000	14
100000	100000	17
1000000	1000000	20





Handout 9 (Runtime)

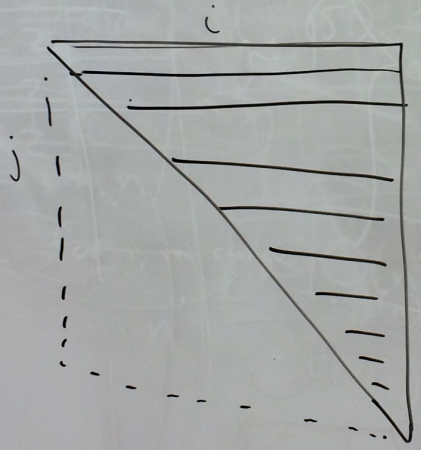
1. n steps
   $O(n)$
   linear

2. 100 steps
   $O(1)$
   constant

3. n + n steps
   (2n)
   $O(n)$

4. $\underbrace{n + n + n + n \cdots n}_{n} \rightarrow O(n^2)$

5. $n + (n-1) + (n-2) + \cdots 3 + 2 + 1$
   (i=0, i=1, i=2)



steps: $\frac{n^2}{2}$

$\Rightarrow O(n^2)$

6. $\underbrace{10 + 10 + 10 \cdots 10}_{n}$
   10n = # steps
   $O(n)$

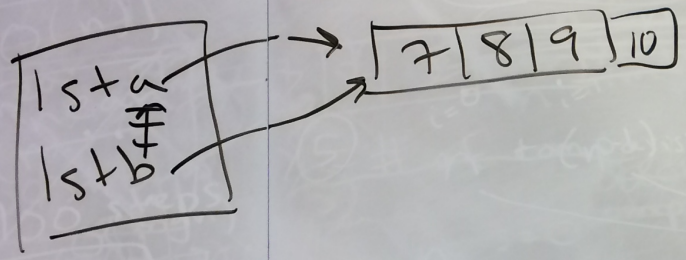
7. $O(\log n)$

8. # steps = 4
   $O(1)$

9. $O(n \log(n))$

Quiz 3 common issues

① stack heap



lsta = [7, 8, 9]

lstb = [7, 8, 9]

lstb = lsta

③
```
def pick_card(low, high):
    cl = randrange(1, 14)
```
low, high+1

⑤
```
if lst[i] < first_word:
    first_word = lst[i]
```
changing

```
x = 3
y = x
y = y + 1
```