CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2017

Swarthmore College

Outline Nov 13:

Right now: start Handout 6 with a partner!

- Recap binary search and runtime: O(log(n))
- Runtime worksheet (Handout 6)
- Begin: sorting

Notes

- Lab 8 due Saturday night (read BEFORE coming to lab!)
- Lab 9 due Monday after Thanksgiving
- Quiz 4 this Friday (let me know if you have conflicts)

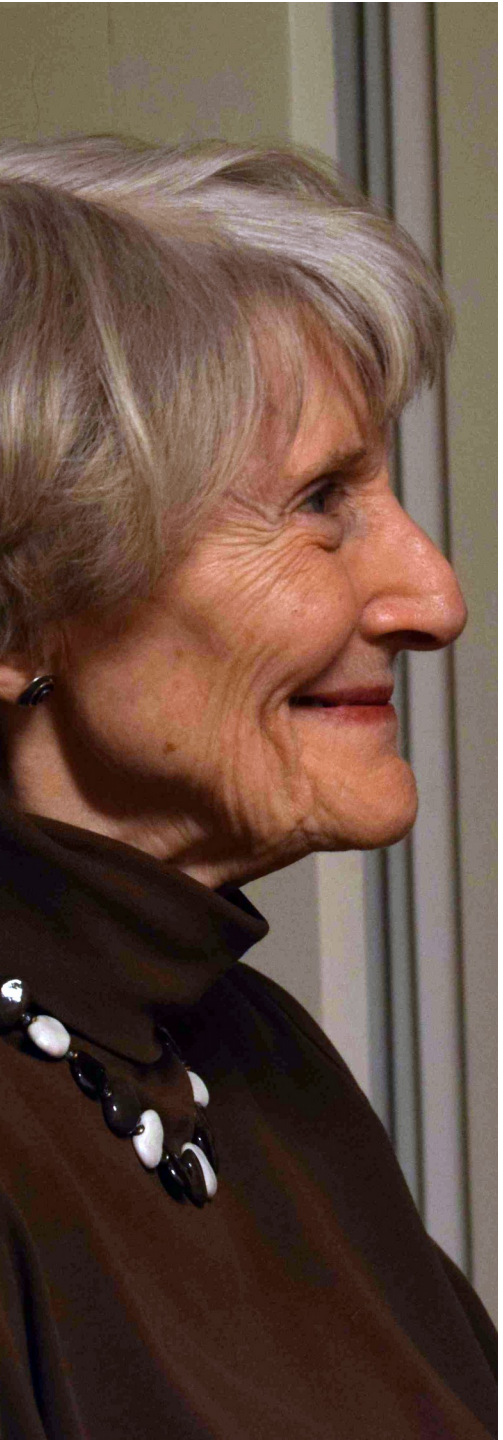
Tips for Lab 8

- **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
```

- **When reading files, so far we have used list accumulators, but you can also accumulate the data as one long string**
- **When in doubt, use an accumulator :)**



Women and the LINC to Modern Computer Technology

A talk by Mary Allen Wilkes (The *first person* to use a personal computer in the home – and designer of the interactive operating system LAP6 for the LINC. Wikipedia)

Wednesday Nov. 15 at 5:00 p.m. Science Center 101

Sponsored by Computer Science, WiCS, and Sigma Xi

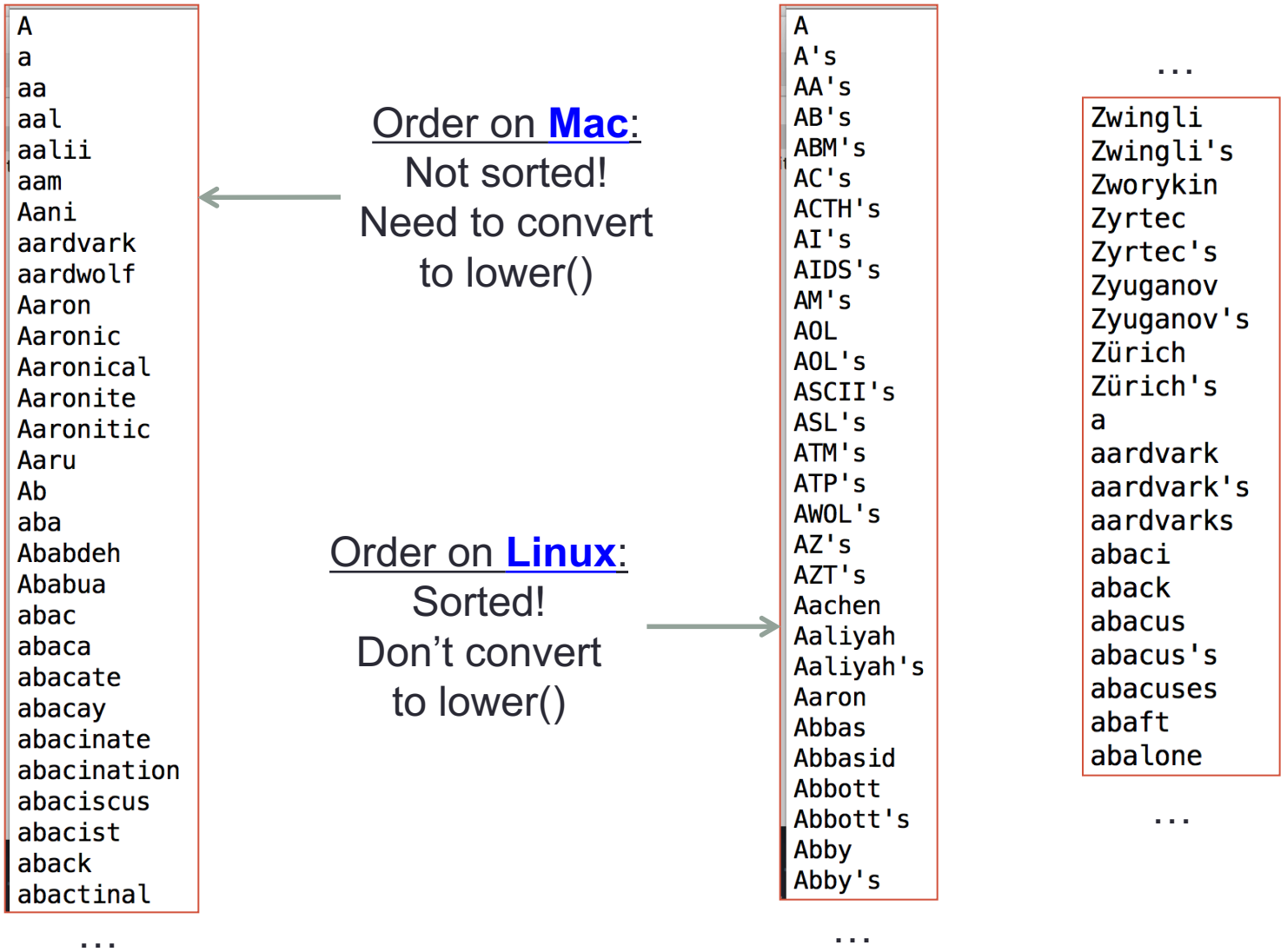
All are welcome!

Binary Search

Binary search: version with comparison counts

```
def binary_search(query, lst):
    low = 0
    high = len(lst) - 1
    count = 0

    # if low surpasses high (low > high), not found: STOP
    while low <= high:
        mid = (low+high)//2
        if query == lst[mid]:
            return mid, count # return both the index and the count
        elif query < lst[mid]:
            high = mid - 1
        else:
            low = mid + 1
        count += 1

    return None, count # return both the index and the count
```

Order of words in dictionary file: (/usr/share/dict/words)



Reading the dictionary file

- For Linux, don't convert to lower! (already sorted)
- All upper case words come before all lower case words in Python as well

```
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
```

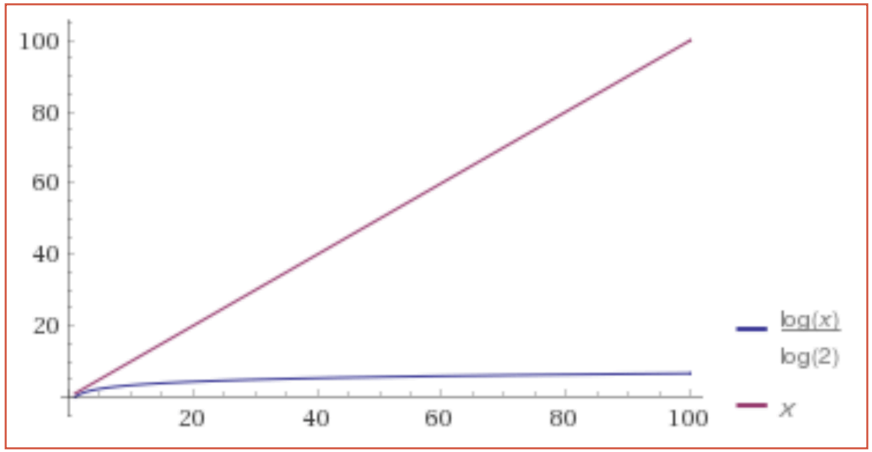
```
>>> 'abba' < 'zip'
True
>>>
>>> 'abba' < 'Abba'
False
>>>
>>> 'Zip' < 'swarthmore'
True
```

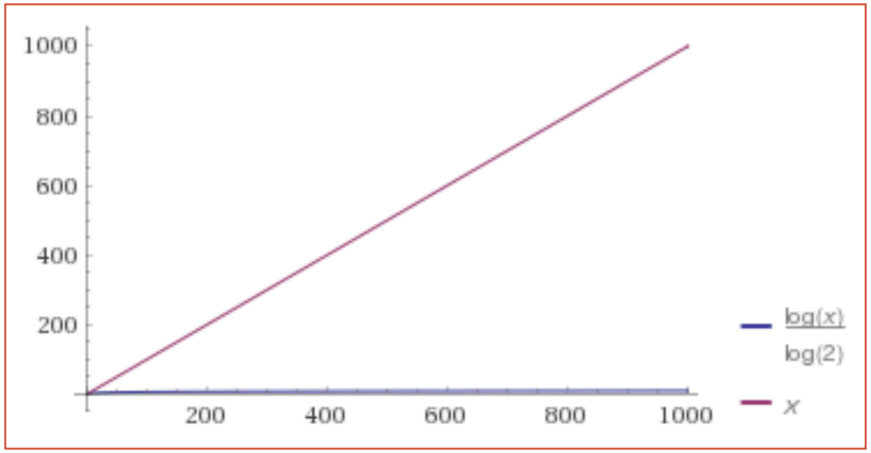
Example of linear vs. binary

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

    n = int(input("Enter the number of words to use: "))
    subset = word_lst[:n] # keep the first n words
    print("first word:", subset[0], "last word:", subset[-1])

    query = input("Enter a word to search for: ")

    print("\nlinear search:")
    index, count = linear_search(query, subset)
    print(query, "is at index", index, "with", count, "comparisons")

    print("\nbinary search:")
    index, count = binary_search(query, subset)
    print(query, "is at index", index, "with", count, "comparisons")

main()
```

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 6 (Runtime)

① for i in range(n):
    print(i)

n steps →

$O(n)$

② for i in range(100):
    print(i*n)

100 steps

constant

$O(1)$

③ for i in range(n):
    print(i)
for j in range(n):
    print(j)

steps: $2n$

$O(n)$

④ for i in range(n):
    for j in range(n):
        print(i, j)

$n + n + \dots n$

n times

$O(n^2)$

⑤ for i in range(n):
    for j in range(i, n):
        print(i, j)

$n + (n-1) + (n-2) + \dots 3 + 2 + 1$

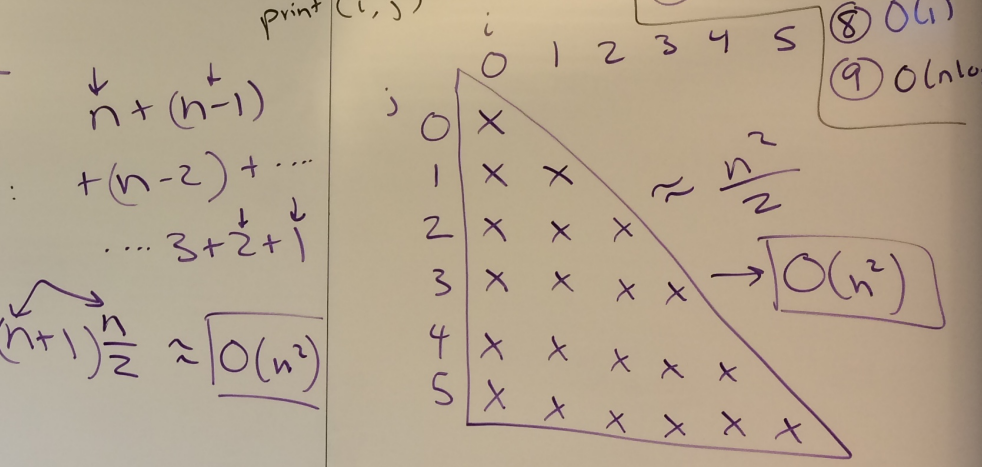
$(n+1)\frac{n}{2} \approx O(n^2)$

⑥ $O(n)$

⑦ $O(\log n)$

⑧ $O(1)$

⑨ $O(n \log n)$



Begin: Sorting

Sorting with cards

1) With a partner, set up a series of ~10 cards (out of order)

2) Try to come up with a **sorting algorithm** that only involves comparing and swapping elements

3) Check your algorithm with me or a ninja

4) Begin implementation in **sorts.py**

5) Here is our swap function from Week 6:

```
def swap(i, j, lst):
    """This function swaps the ith and jth values of the lst."""
    temp = lst[i]
    lst[i] = lst[j]
    lst[j] = temp
```