

## Objectives

- Search strategies: wrap up
- Broader Issue: Social Network Issues

Mar 30, 2012

Sprenkle - CSCI111

1

## Reviewing Lab 10

- Created two classes
  - Used one class within another class
  - Tested them
    - Created .dot file and then a graph of social network
  - Example of a backend to a **real** application
    - Could add a different user interface
- Extension on User Interface: due Tuesday before lab
  - Need to submit electronic version of your code to get the extension

Mar 30, 2012

Sprenkle - CSCI111

2

## Review

- We discussed two different search techniques:
  - What were they?
  - How do they compare?

Mar 30, 2012

Sprenkle - CSCI111

3

## Review: Search Using `in` Review

- Iterates through a list, checking if the element is found
- Known as **linear search**
- **Implementation:**

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

value	8	5	3	7
pos	0	1	2	3

What are the strengths and weaknesses of implementing search this way?

Mar 30, 2012

Sprenkle - CSCI111

search.py

4

## Review: Linear Search

- **Overview:** Iterates through a list, checking if the element is found
- **Benefits:**
  - Works on *any* list
- **Drawbacks:**
  - **Slow**, on average: needs to check each element of list if the element is not in the list

Mar 30, 2012

Sprenkle - CSCI111

5

## Review: Binary Search: Eliminate Half the Possibilities

- Repeat until find value (or looked through all values)
  - Guess middle *value* of possibilities
    - (not middle *position*)
  - 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**

Mar 30, 2012

Sprenkle - CSCI111

6

## Binary Search Implementation

```
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
```

If you just want to know if it's in the list

Mar 30, 2012

Sprenkle - CSCI111

search2.py

7

## 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
    - `__lt__`, `__eq__` methods implemented by the class (or another solution TBA...)
  - List **must** be sorted before searching
    - Takes time to sort

Mar 30, 2012

Sprenkle - CSCI111

8

## Key Questions in Computer Science

- How can we efficiently organize data?
- How can we efficiently search for data, given various constraints?
  - Example: data may or may not be sortable
- What are the tradeoffs?

Mar 30, 2012

Sprenkle - CSCI111

9

## Search Strategies Summary

- Which search strategy should I use under the following circumstances?
  - I have a short list
  - I have a long list
  - I have a long sorted list

Mar 30, 2012

Sprenkle - CSCI111

10

## Search Strategies Summary

- Which search strategy should I use under the following circumstances?
  - I have a short list
    - How short? How many searches? Linear (**in**)
  - I have a long list
    - Linear (**in**) - because don't know if in order, comparable
  - I have a long sorted list
    - Binary

Mar 30, 2012

Sprenkle - CSCI111

11

## Modifying Solution

```
def search(searchlist, key):
    low=0
    high = len(searchlist)-1
    while low <= high :
        mid = (low+high)//2
        if searchlist[mid] == key:
            return mid
        elif key > searchlist[mid]:
            # look in upper half
            low = mid+1
        else:
            # look in lower half
            high = mid-1
    return -1
```

Instead of a list of integers, what if we have a list of Cards and key is a Card object?

- What needs to change?
- What has to be done/verified in the Card class?

Example: player with 2 of clubs starts game of Hearts

Mar 30, 2012

Sprenkle - CSCI111

12

## Card Example

Consider the cards as being in one list

Mar 30, 2012      Sprenkle - CSC1111      13

## Modifying Solution

Instead of a list of integers, what if we have a list of Cards and key is a Card object?

- What needs to change?
- What has to be done/verified in the Card class?

```
def search(searchlist, key):
    low=0
    high = len(searchlist)-1
    while low <= high :
        mid = (low+high)//2
        if searchlist[mid] == key:
            return mid
        elif key > searchlist[mid]:
            # look in upper half
            low = mid+1
        else:
            # look in lower half
            high = mid-1
    return -1
```

Mar 30, 2012      Sprenkle - CSC1111      14

## Comparing Cards

- What are some ways that we could compare Card objects?
- Do we always want one way to compare Cards?

Mar 30, 2012      Sprenkle - CSC1111      15

## Motivating using list's sort method with a key

- We may not want to sort a list of objects by the "standard" way to sort objects
- Consider sorting strings: How does Python sort strings usually?

Mar 30, 2012      Sprenkle - CSC1111      16

## Using list's sort method with a key

- We may not want to sort a list of objects by the "standard" way to sort objects
- Consider sorting strings: How does Python sort strings usually?
  - Alphabetically, upper-case first
- To alphabetize strings, regardless of case:
 

```
words.sort(key=str.lower)
```

↑  
Method to call to do comparison

Mar 30, 2012      Sprenkle - CSC1111      `sort_ignore_case.py`

## Alternative sorting for Card class

- Create a function to use as the key
 

```
def totalOrderCardKeyFunction(card):
    """Returns the key to be used for comparison"""
    # This key means that the cards will be ordered by their
    # rank and then their suit.
    key = "%2d %s" % (card.getRank(), card.getSuit())
    return key
```
- Pass the function name in as the key
 

```
# sort the cards using the key specified by the function
cards.sort(key=totalOrderCardKeyFunction)
```

Mar 30, 2012      Sprenkle - CSC1111      `card.py`

## Extensions to Search

In FaceSpace, we want to find people who are in a certain network.

Consider what happens when **searchlist** is a list of *Persons* and **key** is a network name

- What if we wanted to find a *Person* whose network matched the key and return the *Person*?

Mar 30, 2012

Sprengle - CSCI111

19

## List of Person objects

0	1	2	3	4
Person Id: "1" "George" "Hollywood"	Person Id: "2" "Jennifer" "Friends"	Person Id: "3" "Matt D" "Boston"	Person Id: "4" "Ben A." "Boston"	Person Id: "5" "Lucy L" "Southland"

Example: looking for a person in the "Boston" network...

Mar 30, 2012

Sprengle - CSCI111

20

## List of Person objects

0	1	2	3	4
Person Id: "1" "George" "Hollywood"	Person Id: "2" "Jennifer" "Friends"	Person Id: "3" "Matt D" "Boston"	Person Id: "4" "Ben A." "Boston"	Person Id: "5" "Lucy L" "Southland"

0	1	2	3	4
Person Id: "4" "Ben A." "Boston"	Person Id: "3" "Matt D" "Boston"	Person Id: "2" "Jennifer" "Friends"	Person Id: "1" "George" "Hollywood"	Person Id: "5" "Lucy L" "Southland"

Sorted by network using:

```
personList.sort(key=Person.getNetwork)
```

Mar 30, 2012

Sprengle - CSCI111

21

## Extensions to Solution

```
def search(searchlist, key):
    low=0
    high = len(searchlist)-1
    while low <= high :
        mid = (low+high)//2
        if searchlist[mid] == key:
            return mid
        elif key > searchlist[mid]:
            # look in upper half
            low = mid+1
        else:
            # look in lower half
            high = mid-1
    return -1
```

Consider what happens when **searchlist** is a list of *Persons*

- What if we wanted **all** the *Persons* with the network that matched the key?
- Assumes many different networks

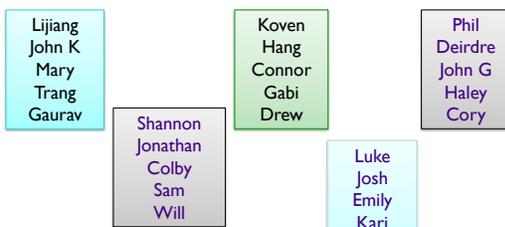
Mar 30, 2012

Sprengle - CSCI111

22

## Broader Issue

- Facebook's News Feed
- Privacy/Security



Mar 30, 2012

Sprengle - CSCI111

23

## Facebook Stats

From a talk by Jeff Rothschild, Vice President of Technology at Facebook in Oct 2009 at UCSD

- Facebook is #2 property on Internet—measured by time users spend on site
- Over 200 billion monthly page views
- >3.9 trillion feed actions processed per day
- Over 15,000 websites use Facebook content
- In 2004, the shape of the curve plotting user population as a function of time showed exponential growth to 2M users. 5 years later they have stayed on the same exponential curve with >300M users.
- Facebook is a global site, with 70% of users outside US

Mar 30, 2012

Sprengle - CSCI111

24

## Broader Issue

- How does Facebook's newsfeed work?
  - What data structures could you use to implement it?
- What are the pros and cons of the News Feed?
  - Anything addressed since the article?
- What are a Social Network's privacy and security issues?
  - How do Facebook/others address these issues?
  - Does knowledge of these issues change your perspective/use of the tools?
- What about Facebook's terms of service?
- Know of any other companies whose algorithms improved business?

Mar 30, 2012

Sprenkle - CSC1111

25

## Discussion

- Good algorithm → Business success
  - Google's PageRank algorithm
    - Revenue: \$16 billion (2007)
  - Facebook's Newsfeed algorithm
    - Revenue: \$150 million/year
  - Others?
- Good algorithm → cost savings
  - Walmart, FedEx
- Good analysis → better usability
  - Travelocity

Mar 30, 2012

Sprenkle - CSC1111

26

## Discussion

- Algorithm uses
  - Lots of data → how is it organized?
  - Fancy frequency tables
    - We have used simplified versions
  - **Data mining, information retrieval**
  - Weight factors (Deal or No Deal offer)
  - **AI** to adapt the weights
- Frequency algorithms, most-recent algorithms: commonly used in OS, Architecture (caching)
- Be careful with Facebook (and MySpace and others) when you're job hunting

Mar 30, 2012

Sprenkle - CSC1111

27

## Next Week

- Tuesday: Finish UI for Social Networking App
  - Continuing development in Tuesday's lab
- Friday: One Laptop Per Child project

Mar 30, 2012

Sprenkle - CSC1111

28