

## Objectives

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

Apr 1, 2011

Sprenkle - CSCI111

1

## Reviewing Lab 10

- Created two classes
  - Used one class within another class
  - Tested them
    - Hopefully created .dot file and then graph to see what you were capable of
- For a **real** purpose
- Extension on User Interface: due Tuesday before lab
  - Need to submit electronic version of your code to get the extension

Apr 1, 2011

Sprenkle - CSCI111

2

## Debugging Note

- I am an excellent debugger
- I have made most of your mistakes
  - Have seen students make the rest
- Program doctor
  - Symptom: Why would X happen?
    - No output, certain error message, printing on separate lines, ...
  - Sometimes need to run some more tests
    - E.g., Print Z ... what additional information does that tell me?
  - Diagnosis: They must have Y!
    - No main() function call, ...

Apr 1, 2011

Sprenkle - CSCI111

3

## 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)

Apr 1, 2011

Sprenkle - CSCI111

4

## Course Evaluations

- Next Wednesday, on Sakai
- General questions about the course
- Specific questions
  - Feedback on improving the broader issues component of the course

Apr 1, 2011

Sprenkle - CSCI111

5

## 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?

Apr 1, 2011

Sprenkle - CSCI111

search.py

6

## Review: Linear Search

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

Apr 1, 2011

Sprenkle - CSCI111

7

## Review

- What was the other search approach?
- What is the algorithm to search that way?

Apr 1, 2011

Sprenkle - CSCI111

8

## 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**

Apr 1, 2011

Sprenkle - CSCI111

9

## 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

Apr 1, 2011

Sprenkle - CSCI111

search2.py

10

## 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 sort

Apr 1, 2011

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 # return True
        elif key > searchlist[mid]: # look in upper half
            low = mid+1
        else: # look in lower half
            high = mid-1
    return -1 # return False
```

What if we had a list of Cards instead of a list of integers and key was a Card?

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

Apr 1, 2011

Sprenkle - CSCI111

12

## Extensions to Solution

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

- What if we wanted to check if the Person's network matched the key and return the *Person*?

```
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]: # look in upper half
            low = mid+1
        else: # look in lower half
            high = mid-1
    return -1 # return False
```

Apr 1, 2011 Sprenkle - CSCI111 13

## Extensions to Solution

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

- What if we wanted **all** the Persons with the network that matched the key?

```
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]:
            # look in upper half
            low = mid+1
        else:
            # look in lower half
            high = mid-1
    return -1 # return False
```

Apr 1, 2011 Sprenkle - CSCI111 14

## Summary of Extensions to Solution

- Check the *network* of the Person at the midpoint
- Get the Persons before and after that Person in the list that have the same network and put in a list
- Represent, handle when no Person matches
- Note: we're not implementing "network contains"
  - How could we implement that?

Apr 1, 2011 Sprenkle - CSCI111 15

## Search Strategies Summary

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

Apr 1, 2011 Sprenkle - CSCI111 16

## Search Strategies Summary

- Which search strategy should I use under the various 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

Apr 1, 2011 Sprenkle - CSCI111 17

## Broader Issue

- Facebook's News Feed
- Privacy/Security

Callie, Jean Paul, Meng

Yates, Ola, Lida, Nick

Will, Anh, Minh

Apr 1, 2011 Sprenkle - CSCI111 18

## 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

Apr 1, 2011

Sprengle - CSCI111

19

## Broader Issue

- How does Facebook's newsfeed work?
  - What data structures would 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/MySpace 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?

Apr 1, 2011

Sprengle - CSCI111

20

## 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

Apr 1, 2011

Sprengle - CSCI111

21

## 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

Apr 1, 2011

Sprengle - CSCI111

22

## Next Week

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

Apr 1, 2011

Sprengle - CSCI111

23