Objectives

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

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

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, …

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)

Course Evaluations

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

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?

search.py

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 -- needs to check each element of list if the element is not in the list

Strategy: 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**

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

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

Modifying Solution

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?

```
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
```


Extensions to Soluti

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

Extensions to Soluti

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
```


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?

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

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

Validating User Input

```
def main():
    #Program mission statement
    print "This program determines your birth year"
    print "given your age and the current year \n"

    age=input("Enter your age: ")
    currentyear=input("Enter the current year: ")

    #Subtract age from current year
    birthyear=currentyear - age
    #Display output to the user
    print "You were either born in", birthyear, "or", \
            birthyear-1
```


Validating User Input

```
def main():
   print "This program determines your birth year"
   print "given your age and the current year \n"

   age=input("Enter your age: ")
   currentyear=input("Enter the current year: ")

   if age < 0 or age > 115:
      print "Come on: that's not a reasonable age."
   elif currentyear < 0:
      print "You need to have a positive year."
   else:
      birthyear=currentyear - age
      print "You were either born in", birthyear, "or", \
            birthyear-1
```

Validating User Input

What happened when the user entered something like “B6”?

Validating User Input

- What happened when the user entered something like “B6”?
  - Threw an **Exception** and the program exited

Python interpreter’s message:

```
Enter your age: B6
Traceback (most recent call last):
  File "currentAge.py", line 22, in <module>
    main()
  File "currentAge.py", line 9, in main
    age=input("Enter your age: ")
  File "<string>", line 1, in <module>
NameError: name 'B6' is not defined
```


Handling Exceptions

- Using try/except statements
- Syntax:

```
try:
        <body>
except [<errorType>] :
        <handler>
```

Optional: use this to handle specific error types appropriately

- Example:

```
try:
        age = input(“Enter your age: ”)
        currentyear = input(“Enter the current year: ”)
except:
        print “ERROR: Your input was not in the correct form.”
        print “Enter integers for your age and the current year”
        return
```


Handling Exceptions

- Could put `try/except` statements in a loop to make sure user enters valid input
  - Example: birthyear3.py
- Other types of exceptions
  - File exceptions:
    - File doesn’t exist
    - Don’t have permission to read/write file

Broader Issue

- Facebook’s News Feed
- Privacy/Security

News Feed:
Kelly Mae
Amy
Andrew
Harrison

Shannon
Dalena
Ben
Hank

Privacy and Security:
Nick
James
Sirocco
Collier
Will

???
Phil
George
Logan
Taylor
Luke
Dave

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

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?
- 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 do you think of Facebook's new look?
- What about Facebook's terms of service?

Discussion

- Good algorithm → Business success
  - Google's PageRank algorithm
    - Revenue: $16 billion (2007)
  - Facebook's Newsfeed algorithm
    - Revenue: $150 million/year
  - Others?

Discussion

- Algorithm uses
  - Lots of data --> how is it organized?
  - Fancy frequency tables
    - We have used simplified versions
  - 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

For Tuesday

- Finish UI for Social Networking App
  - Continuing development in Tuesday's lab