

## Objectives

- Lab 10 Questions?
  - Lab 10 FAQ web page
- Search strategies

Mar 26, 2007

Sprenkle - CS111

1

## Lab 10 Info

- Song
  - writeSong(file)
    - Takes a **file** object (not a filename, which is a string)
    - File object is made in MusicCollection class
- MusicCollection
  - \_\_init\_\_(self, filename)
    - Check if **filename** begins with "libraries/". If not, add that to the **filename**
  - Separate \_\_str\_\_ and displayCollection methods
    - \_\_str\_\_: prints out number of songs, total length

Mar 26, 2007

Sprenkle - CS111

2

## Why getter/accessor methods?

```
class Team:
    def __init__(self, name, mascot, wins, losses):
        self.name=name
        self.mascot=mascot
        self.wins=wins
        self.losses=losses

    def getTotalGamesPlayed(self):
        return self.wins + self.losses

    def getWinPct(self):
        return self.wins/self.getTotalGamesPlayed()

    def getRecord(self):
        return str(self.wins) + "-" + str(self.losses)
```

- Info about the team that you want to know
- Don't know if the Team object keeps that data.
- Do tie games count?

Mar 26, 2007

Sprenkle - CS111

3

## Final Exam Details

- Discuss content a little later
- Give your envelope to me by Thurs, April 3
  - Include our name and proposed time to take the exam on the envelope
- 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)

Mar 26, 2007

Sprenkle - CS111

4

## Course Evaluations

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

Mar 26, 2007

Sprenkle - CS111

5

## Need 4 Volunteers

- No one will get hurt ...

Mar 26, 2007

Sprenkle - CS111

6

## Find the Card in Your Deck

- Reminder to me: take out the jokers
- Challenge: who can find the card first
  - (Most efficient algorithm)
- Need rest of class to keep searchers honest and help me determine who found the card first

Mar 26, 2007

Sprenkle - CS111

7

## The Race is On!

- 3 of Hearts
- 2 of Diamonds
- 4 of Clubs
- Queen of Spades
- King of Queens

Mar 26, 2007

Sprenkle - CS111

8

## Searching for a Playing Card

- Given a deck of cards and a card to find, describe the algorithm for how you would find that card.
  - Present several algorithms (naïve ones too!)
  - Discuss the strengths and weaknesses of each

Mar 26, 2007

Sprenkle - CS111

9

## Search Using **in** Review

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

```
def inSearch(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 26, 2007

Sprenkle - CS111 [search.py](#)

10

## Search Using **in** Review

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

Mar 26, 2007

Sprenkle - CS111

11

## High-Low Game

- I'm thinking of a number between 1-100
- You want to guess the number as quickly as possible (in fewest guesses)
- For every number you guess, I'll tell you if you got it right. If you didn't, I'll tell you whether you're too high or too low

Mar 26, 2007

Sprenkle - CS111

12

## High-Low Game

- I'm thinking of a number between 1-100
- You want to guess the number as quickly as possible (in fewest guesses)
- For every number you guess, I'll tell you if you got it right. If you didn't, I'll tell you whether you're too high or too low

→ What is your best guessing strategy?

Mar 26, 2007

Sprenkle - CS111

13

## Strategy: Eliminate Half the Possibilities

- Repeat until find value (or looked through all values)
  - Guess middle value of possibilities
  - 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 26, 2007

Sprenkle - CS111

14

## Searching for 8

-3	0	0	1	2	7	8	9
0	1	2	3	4	5	6	7

- Find the middle of the list
  - Positions: 0 -- 7, so mid is  $((7+0)/2) = 3$
- Check if the key equals the value at mid (1)
  - If so, report the location
- Check if the key is higher or lower than value at mid
  - Search the appropriate half of the list

-3	0	0	1	2	7	8	9
0	1	2	3	4	5	6	7

8 > 1, so look in upper half

Mar 26, 2007

Sprenkle - CS111

15

## Binary Search

- mid is 5  $((7+4)/2)$ , list[5] is 7

2	7	8	9
4	5	6	7

8 > 7, so look in lower half

- mid is 6  $((7+6)/2)$ , list[6] is 8

8	9
6	7

8 == 8, FOUND IT at position 6!

- What if searched for 6 instead of 8?

Mar 26, 2007

Sprenkle - CS111

16

## Searching for 6

-3	0	0	1	2	7	8	9
0	1	2	3	4	5	6	7

- Will follow same program flow, but 6 is not in the list
- mid is 6, list[5] is 7

2	7	8	9
4	5	6	7

6 < 7, so will try to look in lower half of the list

- mid is 4, list[4] is 2

2
4

2 > 2, so will try to look in upper half of the list, but we've already determined it's not there.  
How do we know to stop looking?

Mar 26, 2007

Sprenkle - CS111

17

## Implementation Group Work

**def** search(searchlist, key):

"""Pre: searchlist is a list of integers in sorted order. Returns the *position* of key (an integer) in the list of integers (searchlist) or -1 if not found"""

- Trace through your program using examples
  - Start simple (small lists)
  - Do what the program says *exactly*, not what you *think* the program says

Mar 26, 2007

Sprenkle - CS111

18

## One 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]:
            low = mid+1
        else:
            high = mid-1
    return -1 # return False
```

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

Mar 26, 2007

Sprengle - CS111

search2.py

19

## 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 search

Mar 26, 2007

Sprengle - CS111

20

## Empirical Study of Search Techniques

- Goal: Determine which technique is better under various circumstances
- How long does it take to find various keys?
  - **Measure** by the number of comparisons
  - Vary the size of the list and the keys
  - What are good tests for the lists and the keys?

Mar 26, 2007

Sprengle - CS111

search\_compare.py

21

## Empirical Study of Search Techniques

- Analyzing Results ...
- By how much did the number of comparisons for **linear search** vary?
- By how much did the number of comparisons for **binary search** vary?
- What conclusions can you draw from these results?

Mar 26, 2007

Sprengle - CS111

search\_compare.py

22

## Broader Issue for Friday

- One Laptop Per Child

Mar 26, 2007

Sprengle - CS111

23