Objectives

- Administration
- Lab 10 Questions?
- Search strategies

Lab 10

- Song
  - writeSong(file)
    - Takes a ***file*** object (not a filename, which is a string)
- MusicCollection
  - readLibrary(filename)
- Extra Credit - submit your own album files for others to use

Demonstrating MyTunes

- Demonstrate “typical” usage of your program
  - User should try out each available option
- You won’t demo command-line args
  - I will test that when I execute your program

Final Exam Details

- Discuss content a little later
- Give your envelope to me by Thursday, December 6
  - Include your name and proposed time to take the exam on the envelope
- In the CS department, all exams are taken in Parmly 405 (our lab)
- At your specified time, someone brings the tests to Parmly 405 and you have 3 hours to take them

Course Evaluations

- Next Wednesday
- General questions (similar to midterm survey)
- Specific questions
  - Feedback on improving the broader issues component of the course

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 “rang in” first)

The Race is On!

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

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 and discuss the strengths and weaknesses of each

Search Using **in** Review

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

value	8	5	3	7
pos	0	1	2	3

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

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

search.py

Search Using **in** Review

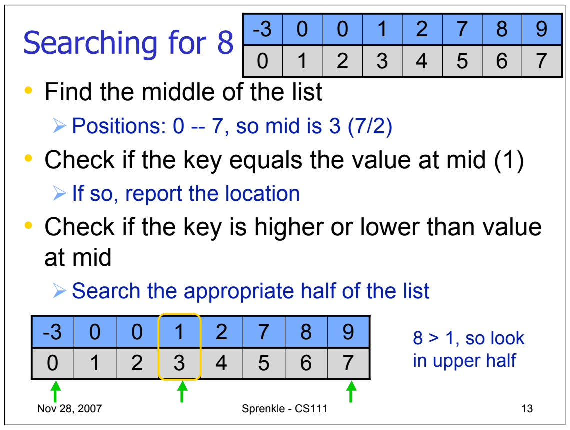
- Iterates through a list, checking if the element is found
- Known as **linear search**
- **Benefits:**
  - Works on *any* list
- **Drawbacks**:
  - Does not tell us where in the list it is
    - What if wanted to do something to that element?
  - Slow -- needs to check each element of list if the element is not found

Binary Search Review

- High-Low game
  - I'm thinking of a number between 1-100
  - You want to guess the number as quickly as possible
  - For every number you guess, I'll tell you whether you're too high or too low or if you got it right
- What is your best strategy?

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 or lower, as appropriate)



Searching for 8

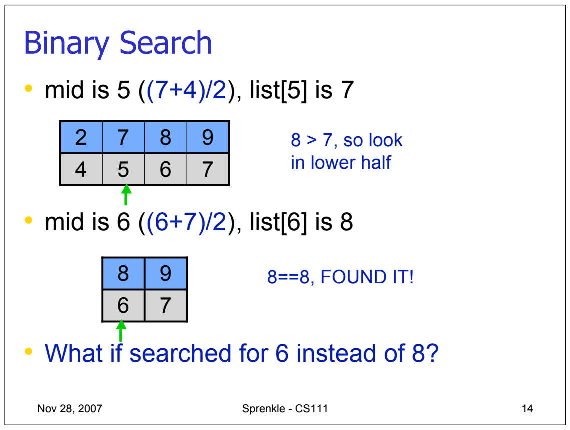
-3	0	0	1	2	7	8	9
0	1	2	3	4	5	6	7

- Find the middle of the list
  - Positions: 0 -- 7, so mid is 3 (7/2)
- 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

Nov 28, 2007 Sprenkle - CS111 13



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

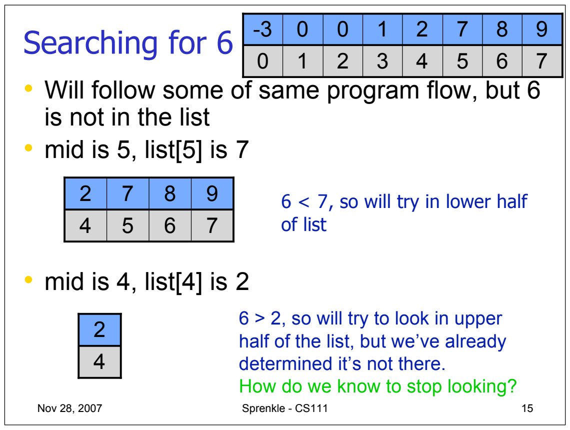
- mid is 6 ((6+7)/2), list[6] is 8

8	9
6	7

8==8, FOUND IT!

- What if searched for 6 instead of 8?

Nov 28, 2007 Sprenkle - CS111 14



Searching for 6

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

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

2	7	8	9
4	5	6	7

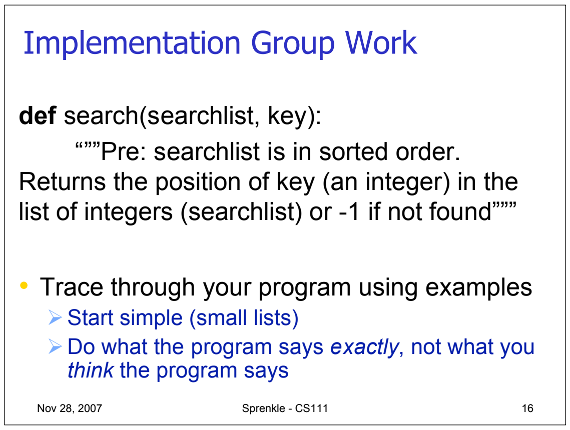
6 < 7, so will try in lower half of list

- mid is 4, list[4] is 2

2
4

6 > 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?

Nov 28, 2007 Sprenkle - CS111 15

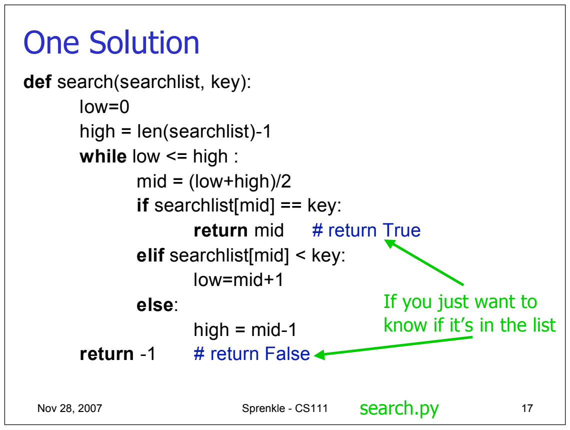


Implementation Group Work

def search(searchlist, key):
    “””Pre: searchlist is 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

Nov 28, 2007 Sprenkle - CS111 16

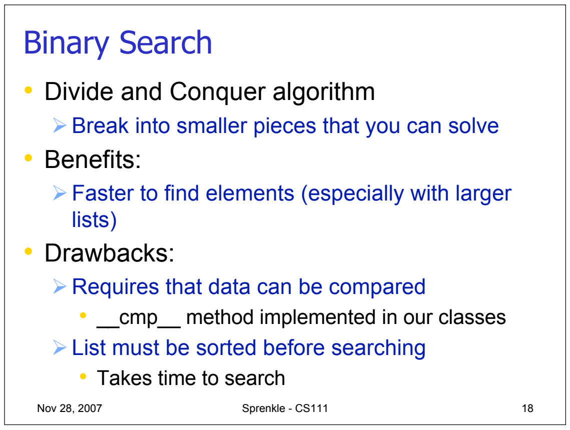


One Solution

```
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 searchlist[mid] < key:
            low=mid+1
        else:
            high = mid-1
    return -1    # return False
```

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

Nov 28, 2007 Sprenkle - CS111 search.py 17



Binary Search

- 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 in our classes
  - List must be sorted before searching
    - Takes time to search

Nov 28, 2007 Sprenkle - CS111 18