

Objectives

- Search strategies

Mar 30, 2011

Sprenkle - CSCI111

1

Lab 10

- Trying to solve a real problem
- Started with designing the solution from a vague specification
- Broke into smaller problems (different classes, different responsibilities)
- Implementing smaller components
- Building to large component

Demonstration of Example UI

Mar 30, 2011

Sprenkle - CSCI111

2

Lab 10 Discussion

- What is the API for the Person class?
- What is the API for the SocialNetwork class?

Mar 30, 2011

Sprenkle - CSCI111

3

APIs

Person

- Person(id)
- str(person)
- getName()
- getNetwork()
- getFriends()
- getNumberOfFriends()
- getId()
- setName(newName)
- setNetwork(newNetwork)
- addFriend(person)

SocialNetwork

- SocialNetwork()
- str(socialNetwork)
- getPerson(id)
- getPeople()
- getUserIds()
- printNetwork()
- addConnection(id1, id2)
- addConnections(filename)
- ...

Mar 30, 2011

Sprenkle - CSCI111

4

Need 2 Volunteers

- No one will get hurt ...

Mar 30, 2011

Sprenkle - CSCI111

5

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 30, 2011

Sprenkle - CSCI111

6

The Race is On!

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

Mar 30, 2011

Sprenkle - CSCI111

7

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 30, 2011

Sprenkle - CSCI111

8

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, 2011

Sprenkle - CSCI111

search.py

9

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

Mar 30, 2011

Sprenkle - CSCI111

10

High-Low Game/TPIR Clock 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

Reminder: write down guesses

Mar 30, 2011

Sprenkle - CSCI111

11

High-Low Game/TPIR Clock 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 30, 2011

Sprenkle - CSCI111

12

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 30, 2011

Sprengle - CSCI111

13

Searching...

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

Use algorithm to search for key = 8

Mar 30, 2011

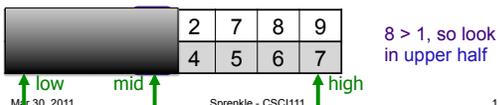
Sprengle - CSCI111

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 position 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



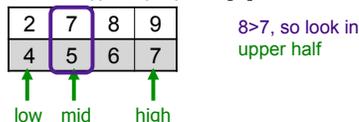
Mar 30, 2011

Sprengle - CSCI111

15

Searching for 8

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



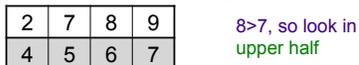
Mar 30, 2011

Sprengle - CSCI111

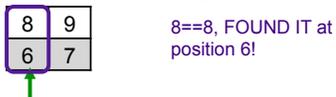
16

Searching for 8

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



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



What if searched for 6 instead of 8?

Mar 30, 2011

Sprengle - CSCI111

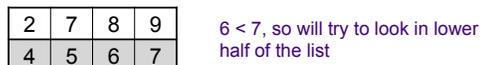
17

Searching for 6

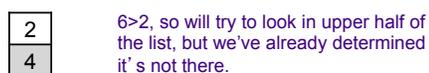
-3	0	0	1	2	7	8	9
0	1	2	3	4	5	6	7

- Will follow same execution flow, but 6 is not in the list

- mid is 6, list[5] is 7



- mid is 4, list[4] is 2



How do we know to stop looking?

Mar 30, 2011

Sprengle - CSCI111

18

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 30, 2011

Sprenkle - CSC1111

19

One Solution

Cutting list in half
Discuss tradeoffs

```
def altBinarySearch(searchlist, key):
    # Base Case: ran out of elements in the list
    if len(searchlist) == 0:
        return NOT_FOUND

    low = 0
    high = len(searchlist)-1
    mid = (low+high)/2

    valueAtMid = searchlist[mid]
    if valueAtMid == key:
        return mid
    if low == high:
        return NOT_FOUND

    if searchlist[mid] < key: # search upper half
        return altBinarySearch(searchlist[mid+1:], key)
    else: # search lower half
        return altBinarySearch(searchlist[:mid], key)
```

Creating a new list
Unnecessary memory use

Mar 30, 2011

Sprenkle - CSC1111

search_divide.py 20

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 30, 2011

Sprenkle - CSC1111

search2.py

21

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)
- Limitations:
 - Requires that data can be compared
 - `__cmp__` method implemented by the class
 - List **must** be sorted before searching
 - Takes time to sort beforehand

Mar 30, 2011

Sprenkle - CSC1111

22

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?

search_compare.py

Mar 30, 2011

Sprenkle - CSC1111

23

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?

search_compare.py

Mar 30, 2011

Sprenkle - CSC1111

24

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, 2011

Sprenkle - CSCI111

25

For Friday

- Broader Issue
 - One of social networking articles
- Lab 10

Mar 30, 2011

Sprenkle - CSCI111

26