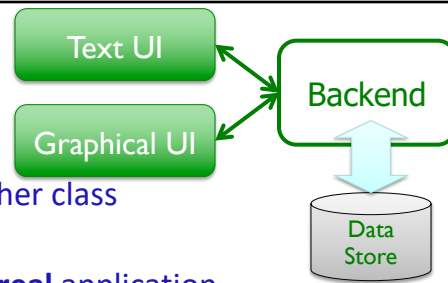


Reviewing Lab 10

- Created two classes
 - Used one class within another class
 - Tested them
 - Example of a backend to a **real** application
 - Could add a different user interface
- “Good judgment comes from experience”
 - Test methods after writing method
 - Remember your data types
 - Refer to the data type’s API
- What could you do to improve your development process?



Mar 29, 2019

Sprenkle - CSCI111

1

Review

- We discussed two different search techniques:
 - What were they?
 - How do they compare?

Mar 29, 2019

Sprenkle - CSCI111

2

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
pos

8	5	3	7
0	1	2	3

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

Mar 29, 2019

Sprenkle - CSCI111

3

Review: Linear Search

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

Mar 29, 2019

Sprenkle - CSCI111

4

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**

Mar 29, 2019

Sprenkle - CSCI111

5

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

Mar 29, 2019

Sprenkle - CSCI111

6

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
 - `__lt__`, `__eq__` methods implemented by the class (or another solution)
 - List **must** be sorted before searching
 - Takes time to sort

Mar 29, 2019

Sprenkle - CSCI111

7

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 29, 2019

Sprenkle - CSCI111

8

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 29, 2019

Sprenkle - CSCI111

9

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 29, 2019

Sprenkle - CSCI111

10

Search Strategies Summary

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

Mar 29, 2019

Sprenkle - CSCI111

11

Search Strategies Summary

- Which search strategy should I use under the following 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
 - Alternatively, may want to sort the list and *then* perform binary search, if sorting first won't be more effort than just sorting.
 - I have a long sorted list
 - Binary

Mar 29, 2019

Sprenkle - CSCI111

12

Extensions to Search

In FaceSpace, we want to find *people* who have a certain name.

Consider what happens when **searchlist** is a list of *Persons* and key is a name (a str)

We want to find a *Person* whose name matches the key and return the *Person*

Mar 29, 2019

Sprenkle - CSCI111

13

List of Person objects

0	1	2	3	4
Person Id:"1" "Gal"	Person Id:"2" "Scarlett"	Person Id:"3" "Chadwick"	Person Id: "4" "Ben"	Person Id: "5" "Samuel"

Example: looking for a person with the name "Chadwick"...

Mar 29, 2019

Sprenkle - CSCI111

14

List of Person objects

0	1	2	3	4
Person Id: "1" "Gal"	Person Id: "2" "Scarlett"	Person Id: "3" "Chadwick"	Person Id: "4" "Ben"	Person Id: "5" "Samuel"

0	1	2	3	4
Person Id: "4" "Ben"	Person Id: "3" "Chadwick"	Person Id: "1" "Gal"	Person Id: "5" "Samuel"	Person Id: "2" "Scarlett"

Sorted by name, e.g.,

```
personList.sort(key=Person.getName)
```

Mar 29, 2019

Sprenkle - CSCI111

15

Extensions to Solution

```
def search(searchlist, key):
    low=0
    high = len(searchlist)-1
    while low <= high :
        mid = (low+high)//2
        if searchlist[mid] == key:
            return mid
        elif key > searchlist[mid]:
            # look in upper half
            low = mid+1
        else:
            # look in lower half
            high = mid-1
    return -1
```

Consider what happens when **searchlist** is a list of *Persons*, **key** is a *str* representing a name
Goal: return a Person object with that name (key)

0	1	2	3	4
Person Id: "4" "Ben"	Person Id: "3" "Chadwick"	Person Id: "1" "Gal"	Person Id: "5" "Samuel"	Person Id: "2" "Scarlett"

Mar 29, 2019

Extensions to Solution

```
def search(searchlist, key):
    low=0
    high = len(searchlist)-1
    while low <= high :
        mid = (low+high)//2
        if searchlist[mid] == key:
            return mid
        elif key > searchlist[mid]:
            # look in upper half
            low = mid+1
        else:
            # look in lower half
            high = mid-1
    return -1
```

Consider what happens when **searchlist** is a list of *Persons*, **key** is a *str* representing the name

Goal: find a *Person* with a certain name

What should we do to make search results more intuitive?

0	1	2	3	4
Person Id: "4" "Ben"	Person Id: "3" "Chadwick"	Person Id: "1" "Gal"	Person Id: "5" "Samuel"	Person Id: "2" "Scarlett"

Mar 29, 2019

Summary of Extensions to Solution

- Check the *name* of the Person at the midpoint
- Represent, handle when no Person matches
- What could we do if more than one person has that name?
- Note: we're not implementing "name contains"
 - How could we implement that?

Mar 29, 2019

Sprenkle - CSCI111

18

How Does Sort Work?

- Several different ways we can sort
- One intuitive way: break down the sort problem into smaller problems
 - Let's say we have a deck of cards that needs to be sorted

Mar 29, 2019

Sprenkle - CSCI111

19

Algorithm: Merge Sort

- I have a list to sort
 - Break the list into two halves
 - Sort the first half
 - Sort the second half
 - Merge those sorted halves together

Mar 29, 2019

Sprenkle - CSCI111

20

Algorithm: Merge Sort

```
def mergeSort( listOfNumbers ):  
    firsthalf = listOfNumbers[:len(listOfNumbers)//2 ]  
    secondhalf = listOfNumbers[len(listOfNumbers)//2:]  
    sortedFirst = mergeSort( firsthalf )  
    sortedSecond = mergeSort( secondhalf )  
    whole = merge( sortedFirst, sortedSecond )  
    return whole
```

But when do we stop calling mergeSort?
Right now, it seems like we will keep calling
mergeSort repeatedly!

Mar 29, 2019

Sprenkle - CSCI111

21

Algorithm: Merge Sort

```
def mergeSort( listOfNumbers ):  
    if len(listOfNumbers) == 2: # base case  
        # sort those two numbers  
        if listOfNumbers[0] > listOfNumbers[1]:  
            temp = listOfNumbers[0]  
            listOfNumbers[0] = listOfNumbers[1]  
            listOfNumbers[1] = temp  
        return listOfNumbers  
    firsthalf = listOfNumbers[:len(listOfNumbers)//2 ]  
    secondhalf = listOfNumbers[len(listOfNumbers)//2:]  
    sortedFirst = mergeSort( firsthalf )  
    sortedSecond = mergeSort( secondhalf )  
    whole = merge( sortedFirst, sortedSecond )  
    return whole
```

Mar 29, 2019

Sprenkle - CSCI111

22

Exam 2 Results

		Section		
	Total	A	B	C
Average	88.50	84.0	85.62	84.1
Median	91.25	87.5	93.55	86.7

Mar 29, 2019

Sprenkle - CSCI111

23

Looking Ahead

- Lab 11

Mar 29, 2019

Sprenkle - CSCI111

24