

Objectives

- Introduction to Recursion
- Comparing Programming Languages

Apr 6, 2022

Sprenkle - CSCI111

1

1

Review: 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
```

Goal: find a Person with a certain name
Consider what happens when **searchlist** is a list of *Persons*, **key** is a *str* representing the *name*

Good capstone problem:

- Brings together
- Algorithms
 - Classes/Objects
 - Lists
 - Methods
 - While loops
 - Strings

0	1	2	3	4
Person Id: "4" "Ben"	Person Id: "3" "Brie"	Person Id: "1" "Gal"	Person Id: "2" "Henry"	Person Id: "5" "Samuel"

Apr 6, 2022

2

2

Solving Binary Search

- Our solution was an *iterative* solution
- We could write it as a *recursive* solution
- **Recursion**: method of solving problems
 - Break a problem down into smaller subproblems of *the same problem* until problem is small enough that it can be solved trivially

Apr 6, 2022

Sprenkle - CSCI111

3

3

Toward Recursive Binary Search

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

Apr 6, 2022

Sprenkle - CSCI111

4

4

Toward Recursive Binary Search

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

The function calls itself on a list that is ~half the size of the original!



Recursion: Breaking problem into smaller subproblems of the same problem ... into trivial solution

Apr 6, 2022

Sprenkle - CSCI111

5

5

Toward Recursive Binary Search

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

The function calls itself on a list that is ~half the size of the original!



When does the function stop?
What is the trivial solution we're trying to break down to?

Apr 6, 2022

Sprenkle - CSCI111

6

6

Toward Recursive Binary Search

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

← Stops when we find the element

But, what if the element isn't in the list?
When will we know that?

Apr 6, 2022

Sprenkle - CSCI111

7

7

Recursive Binary Search

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

← Stops when we find the element

But, what if the element isn't in the list?
When will we know that?

Apr 6, 2022

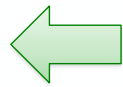
Sprenkle - CSCI111

8

8

Recursive Binary Search

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



Base case: We know the key is not in our list

Apr 6, 2022

Sprenkle - CSCI111

9

9

Recursive Binary Search Conclusions

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

- Broke problem into smaller problems
- Smallest problem is easy to solve
- BUT, this is not an efficient solution because creates multiple lists
 - (Can write a recursive solution that doesn't create multiple lists but would need to change the function signature)

Apr 6, 2022

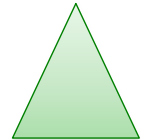
Sprenkle - CSCI111

10

10

Recursion Summary

- **Recursion**: method of solving problems
 - Break a problem down into smaller subproblems until problem is small enough that it can be solved trivially
- **Binary Search**:
 - Break problem to ~half the size of original problem
 - Base cases: when the middle element is what you're looking for; when there are no elements in your list
- Any recursive problem can be solved iteratively
 - Some problems lend themselves better to recursive solutions



COMPARING PROGRAMMING LANGUAGES

Applying What You Know To Other Languages

- At the beginning of the semester, some of you wondered
 - “Why the Python programming language?”
 - “Will I be able to read/write programs in other programming languages?”
- We’ll answer the first question by showing that you can do the second

Apr 6, 2022

Sprengle - CSCI111

13

13

Applying What You Know To Other Languages

- **Syntax:** symbols used
- **Semantics:** what the symbols *mean*

Apr 6, 2022

Sprengle - CSCI111

14

14

What is the Python 3 Program Doing?

Apr 6, 2022

Sprenkle - CSCI111

15

15

What is the Python3 Program Doing?

- Getting a line of input from “standard in” (from the user)
- Splitting the input into integers
- Calculating the result of a formula
- Deciding if a student is admitted, based on the result of the formula
- Displaying the result

Apr 6, 2022

Sprenkle - CSCI111

16

16

Admissions Problem

- Binary University decides to admit students based on a formula that weighs various factors
 - Scores of 70 or better are admitted
- Input: single line, 4 integers, in order below

Category	Range	Weight Factor (Multiplier)
AP Courses	0-10	3
Intangibles	1-10	2
High School GPA	0 - 100	0.25
SAT score	400-1600	.02

Apr 6, 2022

Sprenkle - CSCI111

17

17

Example Input/Expected Output

Input	Expected Output
0 1 0 300	DENY
6 10 99 1590	ADMIT
0 7 82 1500	ADMIT
2 5 80 990	DENY
5 5 92 1200	ADMIT
2 5 100 1300	ADMIT

Apr 6, 2022

Sprenkle - CSCI111

18

18

What is the Python3 Program Doing?

- Getting a line of input from “standard in” (from the user)
- Splitting the input into integers
- Calculating the result of a formula
- Deciding if a student is admitted, based on the result of the formula
- Displaying the result

Apr 6, 2022

Identify these pieces in the other programs

19

19

Comparing Programming Languages

- How is the syntax/semantics of these languages different from Python?
- What is easier or harder to do in these other programming languages than in Python?

Apr 6, 2022

Sprenkle - CSCI111

20

20

Comparing Programming Languages

Benefits of Python

- Simpler syntax (e.g., fewer `{}` and `()`)
- Can cover some content with less overhead

Drawbacks

- Data types aren't explicit (static)
 - Can be harder for you to remember and keep straight
- Not compiled explicitly beforehand
 - Keep executing to find all the syntax bugs
 - Doesn't check: "you're passing a file instead of a string"
- Allows you to do some things that won't work in other programming languages

Apr 6, 2022

Sprenkle - CSCI111

21

21

Bash

- Scripting language
 - Can call Unix commands
- Example program:
 - `createPrintableLab`

Apr 6, 2022











Sprenkle - CSCI111

22

22

Tiobe Index

based on the number of skilled engineers world-wide,
courses and third party vendors

Mar 2022	Mar 2021	Change	Programming Language	Ratings	Change
1	3	CSCI111, 112	 Python	14.26%	+3.95%
2	1	CSCI210, 320	 C	13.06%	-2.27%
3	2	CSCI209, 335	 Java	11.19%	+0.74%
4	4		 C++	8.66%	+2.14%
5	5		 C#	5.92%	+0.95%
6	6		 Visual Basic	5.77%	+0.91%
7	7	CSCI335	 JavaScript	2.09%	-0.03%
8	8		 PHP	1.92%	-0.15%
9	9		 Assembly language	1.90%	-0.07%
10	10	CSCI335, 317	 SQL	1.85%	-0.02%

Apr 6, 2022

http://www.tiobe.com/tiobe_index

23

23

Final Exam

- Final will be in Canvas
 - Take anytime during finals (Saturday a.m. – Friday at noon)
 - Due end of exam period - Friday at noon
- Prep document on schedule
 - Similar format to previous exams but in Canvas
 - More on Friday

Apr 6, 2022

Sprenkle - CSCI111

24

24

Looking Ahead

- Thursday: BI write up due
- Friday:
 - Lab 11 due
 - Review computer science
 - Where we've been and where you can go
 - Bring your exam questions
 - Practice
- All (late) lab work and extra credit articles must be submitted by **MONDAY 11:59 p.m.**