CIS 210, Fall 2016
Introduction to Computer Science

Main Menu

- Class home page
- Piazza
- How to Succeed
- Schedule
- Assignments
- References
- Exams

Project 5, Part 2

This assignment is due Friday, October 28 at 5pm. Save your Python program in a file called is_member.py and turn that file in using Canvas.

Purpose

In this project you will need to design and implement a recursive algorithm for determining set inclusion. It is not a long or complicated project, but it requires careful design.

Pair Assignment

Your are encouraged to use *pair programming* to complete this assignment. Work together with one classmate. The amount of actual programming required for this project is small, but be sure you understand how the main program logic works, including the recursive function.

Sets

A set is a well-defined collection of *distinct* objects (there are no duplicates). The objects that make up a set (also known as the elements or members of a set) can be anything: numbers, people, letters of the alphabet, other sets, and so on.

Membership

If B is a set and x is one of the objects of B, this is denoted x ∈ B, and is read as "x is an element of B".

Set representation

For this assignment, we will represent a set of integers as a sorted list of the unique integers. This is different from Python's built-in set construct, in which the elements are unordered. By using a sorted list, you will be able to determine whether a value is a member of a given set by using a recursive algorithm.

Binary search

In computer science, a binary search algorithm finds the position of a target value within a sorted array. The binary search algorithm begins by comparing the target value to the value of the middle element of the sorted array. If the target value is equal to the middle element's value, then the position is returned and the search is finished. If the target value is less than the middle element's value, then the search continues on the lower half of the array; or if the target value is greater than the middle element's value, then the search continues on the upper half of the array. This process continues, eliminating half of the elements, and comparing the target value to the value of the middle element of the remaining elements - until the target value is either found (and its associated element position is returned), or until the entire array has been searched (and "not found" is returned).

Example

Sorted array: l = [1, 3, 4, 6, 8, 9, 11]
Target value: x = 4
Compare x to 6; x is smaller, so repeat with [1, 3, 4]
Compare x to 3; x is larger, so repeat with [4]
Compare x to 4; x == 4, so the position is returned

Requirements

Your program will read a file containing a list of integers, which may be in any order; you are guaranteed that there are no duplicate values in the file. Each integer appears on a line by itself, and every line in the file contains an integer (there are no blank lines). Your program will create a list of integers from those contained in the file, and will have to sort them to convert the list into our representation of a set.

Besides reading the set of integers from the file, you must also implement the is_member() function, which given a set represented as a sorted list, and a candidate number, returns True if that number is an element of the the set, and False otherwise. Your implementation must recursively implement the binary search algorithm described above, with the exception that we do not need to know the index for the matched element in the list, we simply need to know True or False. When your is_member() function recursively invokes itself, the list it passes on the recursive call should be a slice of the original list.

Your program will be invoked from the command line as follows:

```
$ python3 is_member.py elements.txt number
```

Getting started

This starter code may be helpful. You will need to create an elements.txt file containing integers that make up a set against which you test your implementation. You will also need to access the test_harness.py found here.

Grading

40 points possible

- 20: Program runs, correctly implements is_member() using a recursive implementation. 5 points if it runs, but incorrectly implements is_member(). 0 points if it does not run.
- 10: Follows CIS 210 coding guidelines, including author identification and header .
- 10: Clarity. The program should not only be consistent with the requirements and approach described here, but it should be very easy to read the program and verify its consistency with the spec.