Audit Access Expires 5 de ago de 2020
You lose all access to this course, including your progress, on 5 de ago de 2020.

Exercise 4

Finger Exercises due Aug 5, 2020 20:30 -03

Exercise 4

1/1 point (graded)

ESTIMATED TIME TO COMPLETE: 5 minutes

Here is some code for linear search that uses the fact that a set of elements is sorted in increasing order:

```
def search(L, e):
    for i in range(len(L)):
        if L[i] == e:
            return True
        if L[i] > e:
            return False
    return False
```

Consider the following code, which is an alternative version of `search`.

```
def search3(L, e):
    if L[0] == e:
        return True
    elif L[0] > e:
        return False
    else:
        return search3(L[1:], e)
```

Which of the following statements is correct? You may assume that each function is tested with a list `L` whose elements are sorted in increasing order. For simplicity, assume `L` is a list of integers.

- ◯ `search` and `search3` return the same answers.
- ◯ `search` and `search3` return the same answers provided `L` is non-empty.
- ◉ `search` and `search3` return the same answers provided `L` is non-empty and `e` is in `L`.
- ◯ `search` and `search3` do not return the same answers.
- ◯ `search` and `search3` return the same answers for lists of length 0 and 1 only.

✔

Explanation:
`search3` is a recursive function. It will return the correct answer... provided that `L` is non-empty and `e` is in `L`! Why is this? Consider the first line of code in the function:

```
if L[0] == e:
```

If `L` is an empty list, this will throw an error, because `L[0]` does not exist. If `e` is not in `L`, this same line will also throw an error - because we will walk through every element in `L`, and eventually `L` will be an empty list! If you're having trouble seeing this, try running this version of `search3`:

```
def search3(L, e):
    print("List L: " + str(L))
    if L[0] == e:
        return True
    elif L[0] > e:
        return False
    else:
        return search3(L[1:], e)
```

Run the following two calls and watch the print out carefully.

```
search3([], 4)
search3([1, 2, 3], 4)
```

How would you change search3 to avoid this problem?

The easiest way to modify `search3` to avoid this problem would be like this:

```
def search3(L, e):
    # Test if the list is empty - if it is, e cannot be in it!
    # Run this test first - so that we don't throw an error trying
    #  to access L[0].
    if L == []:
        return False

    if L[0] == e:
        return True
    elif L[0] > e:
        return False
    else:
        return search3(L[1:], e)
```

Enviar

Answers are displayed within the problem

Exercise 4

Ocultar discussão

Topic: Lecture 12 / Exercise 4

Add a Post

Show all posts por atividade recente

There are no posts in this topic yet.