15-110: Principles of Computing, Spring 2018

Lab 4 – Thursday, February 8

Goals

- Review how to perform a simple linear search and how to time a function.
- Analyze linear search by timing it for various searches to find patterns.
- Modify linear search so it can solve a related problem.
- Use a while loop to search a "gapped" list – that is, every nth element of a list.

Part 1 : Linear Search and Timing Code [TA Demonstration]

1.1 Creating a List Containing a Range of Elements

```
>>> list(range(0,10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

1.2 Linear Search and a Timing function

Create a `lab4` directory. In this directory, store the file `timer.py` from Autolab. In `timer.py`, the following function finds the index of the first occurrence of the `key` in `datalist`:

```
def search(datalist, key):
    for i in range(0, len(datalist)):
        if datalist[i] == key:
            return i
    return None
```

In the same file `timer.py`, the following function measures the time it takes to run this search for a list of integers from 100,000 through 300,000, inclusive.

```
import time
def timer(key):
    biglist = list(range(100000, 300001))
    start = time.time()
    search(biglist, key)
    stop = time.time()
    return stop - start
```

Part 2: Student Activities

2.1 Timing

Create a file `timing.txt` using gedit. In `timing.txt`, write your answers to the following problems.

a. Time how long the `search` function above takes to find each of the following in `biglist`:

   i. 100,000 (element at the front of `biglist`)
   ii. 200,000 (element at the center of `biglist`)
   iii. 300,000(element at the end of `biglist`)
   iv. 3 (element not in `biglist`)

b. If you repeat the same searches, does it take the same amount of time? Why?

c. Which searches depend on the length of the list? Which do not?

2.2 Are they all odd?

Recall we can use modulo to determine if an integer is odd or even:

```
>>> 4 % 2    # 4 is even
0
>>> 5 % 2    # 5 is odd
1
```

In your `lab4` directory, create a file `is_all_odd.py` using gedit. In `is_all_odd.py`, define a Python function `is_all_odd(numlist)` that takes an integer list as an argument and returns `True` if all elements in `numlist` are odd numbers and `False` otherwise.

Algorithm: Modify the linear search algorithm (1.2). For each index `i`, if `datalist[i]` is even, then the list can't be all odd so return `False`. If you searched through the whole list (i.e. tried every index `i`) and never saw an even element, then return `True` at the end of the function.

Example Usage:

```
-bash-4.2$ python3 –i is_all_odd.py
>>> is_all_odd([1, 2, 8, 12, 99])
False
>>> is_all_odd([1, 3, 5, 7])
True
>>> is_all_odd([11])
True
>>> is_all_odd([])
True
```

2.3 Gapped List

In this exercise, you will iterate over a "gapped" list (i.e., by examining every n^{th} element). In your `lab4` directory, create a file `gap_search.py` using gedit. In `gap_search.py`, write a Python function `gap_search(datalist, key, gap)`. This function works like the `search(datalist, key)` from 1.2 above, but with these important differences:

a. You will use a `while` loop instead of a `for` loop; and
b. The search must start with the first element, then skip over the number of elements given by `gap`, and so on. You may assume gap is greater than or equal to 1. For example, if we have a `gap` of 2 and the following datalist: `['t','u','v','w','x','y','z']`,then the search must examine `'t'`, `'v'`, `'x'`, and `'z'`. However, if the `gap` is 3, then the search must examine `'t'`, `'w'`, and `'z'`; if the `gap` is 4, it must examine `'t'` and `'x'`; and if the `gap` is greater than 6, only `'t'` must be examined.

Use the following algorithm:

1. Set `i` equal to 0.
2. While `i` is less than the length of `datalist`, do the following:
    a. If the element at index `i` in `datalist` is equal to `key`, then return `i`.
    b. Otherwise, add `gap` to `i`.
3. If you get here, the key is not found, so return `None`.

Example usage:

```
-bash-4.2$ python3 –i gap_search.py
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'x', 1)
4
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'x', 2)
4
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'x', 3)
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'x', 4)
4
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'x', 5)
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'x', 6)
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'x', 7)
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'a', 1)
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'z', 1)
6
>>> gap_search(['t', 'u', 'v', 'w', 'x', 'y', 'z'], 'z', 3)
6
```

Submission

When you finish the lab, you should be inside the `lab4` folder, which is inside the directory `private/15110`. When you type '`ls`' and press the `Enter` key, you should see the following files: **`timer.py, timing.txt, is_all_odd.py,`** and **`gap_search.py`**. Once you see all files, please type '`cd ..`' and press the `Enter` key. Then, zip your `lab4` folder by typing: `zip –r lab4.zip lab4`
You will see the files being compressed. Then submit the zipped file `lab4.zip` on Autolab.