Review Questions and Study Guide

Searching, Sorting and Efficient Algorithms

Reference: Lab 6, Lecture 6, Assignment 6.

Key ideas: Sequential search (starting at the beginning of a sequence and proceeding through the sequence until you find what you're looking for); binary search; simple sorting algorithms like insertion sort; sophisticated sorting algorithms like merge sort. How long do each of these take as a function of the size of the list?

Representative Problems:

1. Suppose you search for an object \( x \) in an unsorted sequence \( s \) that has 10,000 items, by using

\[
\text{s.find}(x)
\]

How many comparisons are done between \( x \) and items in \( s \) are performed? (This will vary—you could get lucky and find \( x \) on the very first try. But what is the worst case? How long does it take on average?)

2. Same question, however now the sequence \( s \) is sorted and you use binary search. (Don't worry about the average—it's a little hard to figure out, but it turns out to be very similar to the worst case.)

3. Now suppose we have two sequences \( s_1 \) and \( s_2 \), the first with 10,000 items and the other with 100,000 items. We want to find out how many items in \( s_2 \) are contained in \( s_1 \). How many comparisons of items in \( s_2 \) with items in \( s_1 \) will we perform if \( s_1 \) is unsorted? If \( s_1 \) is sorted?

4. What if both \( s_1 \) and \( s_2 \) are sorted in the above problem?

Bits and bytes

Reference: Lab 9, Lecture 8


Representative Problems: All the problems from Lab 9! Here are a few others, more centered around general understanding than actually working with the representations.
1. How many bytes are required to store the text of a novel in ASCII. Depends on the novel, of course! But open a book--how many pages does it have? Estimate the number of characters on a page, and figure out the number of characters in the book.

2. I back up my computer with a 2TB hard drive. *Exactly* how many bytes are 2TB? (It's not 2 trillion.) How many novels could I store in this drive?

3. We studied the binary representation of colors using three bytes to represent a color. How many different colors are there using this scheme?

4. The ASCII code for lower-case letter *a* is 97 in decimal. What is the output when you execute

   ```python
   print('\x63\x61\x74')
   ```

   If we used the same 3-byte sequence to represent a color, what would that color look like?

## Multidimensional lists; image processing

Reference: Assignment 7

**Representative Problem:** Below are four functions designed to alter an image:

```python
def a(im):
    height=len(im)
    width = len(im[0])
    newim=[
        for row in range(height):
            newcol =[
                for col in range(width):
                    newcol.append([0,0,0])
            newim.append(newcol)
        for row in range(height):
            for col in range(width):
                newim[row][col]=im[(row+200)%height][(col+200)%width]
    return newim

def b(im):
    height=len(im)
    width = len(im[0])
    newim=[
        for row in range(height):
            newcol =[
```
for col in range(width):
    newcol.append([0,0,0])
newim.append(newcol)
for row in range(height):
    for col in range(width):
        newim[row][col]=im[height-row-1][width-col-1]
return newim

def c(im):
    height=len(im)
    width = len(im[0])
    for row in range(height):
        for col in range(width):
            im[row][col]=im[height-row-1][width-col-1]
    return im

def d(im):
    height=len(im)
    width = len(im[0])
    newim=[]
    for row in range(height):
        newcol =[]
        for col in range(width):
            newcol.append([0,0,0])
        newim.append(newcol)
    for row in range(height):
        for col in range(width):
            newim[row][col]=im[col%height][row%width]
    return newim

When we apply these to our clown image, we get the following results, although not in the same order. Match the code to the resulting image.
Tuples, dictionaries, and sets

Reference: Lab 10; lecture slides and lecture recordings on this subject, Assignment 8.

Key ideas: This is mostly about coding, mastering the syntax of tuples, dictionaries and sets and being able to manipulate them with confidence. There are also some underlying ideas about efficient manipulation: When you use a dictionary, you can retrieve the value associated with a key very rapidly, where the time does not depend on the size of the dictionary, and with no need to examine other items in the dictionary.

Representative Problems: The problems from Lab 10. Also see the posted midterm from Fall, 2018. Here is one more problem:

Redo the `find_anagrams` function from Assignment 6, using Python dictionaries. That is, take the original word list, and construct a dictionary in which a typical item looks like:

```
'acert':['cater','crate','react','trace']
```

Compare the performance of this implementation with the two versions of `find_anagrams` from the assignment, both the time required to construct the dictionary (or list in the prior versions), and the time to recover the anagrams for a word.