Columbia College
Vancouver, Canada

Introduction to Computer Science and Programming 1 – CSCI120

Chapter 13: Algorithm Complexity Analysis

Sample Coding / Problems

Problem1

- Define complexity class of each of the following algorithms (operations) – Complete the table below:
  1. Adding an item to a set
     a. O(1) or constant
  2. Checking whether the item already exists in the set or not
     a. O(1) or constant
  3. Updating the value of a specific key in the dictionary
     a. O(1) or constant
  4. Removing an item from the dictionary
     a. O(1) or constant

Problem2

- Think about a solution for each problem and then figure out the time complexity order and the class of complexity for the proposed solution:

1. We want the ask user to enter words as long as the user has not entered the word "exit". We would like to know how many words have length more than 5.

Assumption: Checking the len(word) is O(1)
n*O(1) = O(n)

2. We are given a number and we would like to reverse it.

Rev = ""
321
321//10 = 32
321%10 = 1 *
rev = rev + "1" -> rev = 1
32//10 = 3
32%10 = 2 *
rev = rev + "2" -> "1"+"2" = "12"
3//10 = 0
3%10 = 3
Rev = rev + "3". -> "12" + "3" -> 123

n = number of digits in the number
time complexity = o(n)

3. Number a and number b are given and we would like to calculate a to the power of b. (a**b).

a = 4
b = 30
2^3 = 2*2*2*….*2
Time complexity -> O(b)

4. A list of numbers is given and we would like to know whether it is sorted or not.

[2,3,6,5,8,9,1]
Items[i] <= items[i+1]. -> asc sorted
Items[i]>= items[i+1] -> des sorted

n = len(list) -> O(n)

Problem3

- What data structure would you use in order to represent the following pieces of information.

1. A receipt of a purchase which includes the code of the purchased products and their expenses.
    - code: string
    - price: float/int
        - dictionary -> {code, price} , class (purchase) [instance properties: code, price]
    - receipt -> list of dictionary {code, price}
    - receipt -> list of object of purchase
2. To represent the keyboard of your macbook or widnows laptop (assume the key's sizes are equal)
    - key: list / set
    - list of keys is not very accurate
        - 2D list of keys
        - 2D list of lists (or set)
3. To represent the courses you have taken so far on your transcript.
    - Course -> dictionary , class course (code, name, ..)
    - Transcript List of courses
    - If you want to show the semesters and coruses in each semester:
        - Dictionary : key: semester no -> value: list of courses
4. To represent the contact list of your mobile phone
    - Contact list -> name: number
        - Dictionary: key is name, value: number (if the names are unqiue)
        - 2 lists
            - List of names
            - List of numbers
        - Tuple (name, number) -> list of tuples

Columbia College
Vancouver, Canada

Problem4

Suppose you would like to know how many people are living in a high-rise building. There are 15 floors in this building. There is a dictionary which shows how many people are living in each floor. The building does not have floor 13. Here is the list of floors and number of people in each floor:

```
floorDictionary = {1:6 , 2:4, 3:10, 4:8, 5:6, 7:4, 8:8, 9:5, 10:4, 11:4, 12:4, 14:4, 15:1}
```

In the above dictionary the key is the floor number (which is a number) and the value is number of people living in that floor.

```
def getPeopleInFloor(floorNumber):
    if floorNumber in floorDictionary:
        return floorDictionary[floorNumber]
    else:
        return 0
```

The above function receives the floor number and returns number of people living in the floor. Now use the above function to:

- Write another function (non-recursive) which receives a list of floor numbers and returns the total number of people living in this building.
- Write a recursive function which receives a list of floor numbers and returns the total number of people living in this building

Good Luck ☺