CG1112 Engineering Principle and Practice II

Semester 2 2020/2021

Week of 22nd January 2020

Tutorial 1

Pi, Git and Complexity

Tutorial in CEG EPP II is used to consolidate learning points from the studio(s). Please refer to the respective studio handouts and online materials before attempting the questions. You will get the most benefit by working out the solution before the tutorial session.

1. [Raspberry Pi] You have now setup and used a Raspberry Pi 3B+. Build a table and contrast the RPi with a typical laptop computer in the following areas:
    a. Power Requirements (Voltage, Current for typical usage).
    b. Hardware Specification (CPU clock speed, cores, and runtime memory).
    c. Storage (SD Card vs. HDD vs SSD - read/write speed)
    d. Interfacing capabilities (to other devices, components, networking etc).
    e. Software environment

    Based on the above, suggest 1-2 scenarios where Pi is more suitable than a laptop or desktop.

    In coming studios, you will learn about the Arduino Uno, and then you can further contrast the RPi with that device.

2. [Algorithm Choice Matters] Let's consider the sum of the first 100 integers.

    1 + 2 + …. + 98 + 99 + 100 = 5050

    Describe or write out the simplest and most obvious algorithm to calculate this sum for the first N integers. What is the time complexity of your method? You can assume addition takes constant time O(1). If we double the problem size, how much more time does it take?

    As a high school student, the mathematician Carl Fredrich Gauss, impressed his teacher by finding the sum of the integers from 1 to 100 very quickly. He used a different algorithm:
    Gauss realised he had fifty pairs of numbers when he added the first and last number in the series, the second and second-to-last number in the series, and so on, and that each of these had the same size. For example (1+100), (2+99), and so on. All of these total to 101. Therefore, we know the sum is 101*50 or 5050. You can assume that multiplication takes O(1) time.

What is the time complexity of this method? If we double the problem size, how much more time does it take?

3. [Complexity] Give the time complexity of the workload functions from Week 2 – Studio 1 using Big-O notation:
   a. WorkA(54321 * N);
   b. WorkB(73 * N);
   c. WorkC(5 * N);
   d. WorkE(N);
   e. [Beyond Scope] WorkD(N);

4. [Time & Space Complexity] Consider the following problem:

Given a character strings of N characters, tally the frequency of occurrences for every characters and print out the answer.

Example: "**ab!da!**" (N = 6 characters)
Output:

```
a = 2 times
b = 1 time
! = 2 times
d = 1 time
a = 2 times          //note the result is printed for every characters in the
! = 2 times          // input string, regardless of duplication.
```

Suggest **two algorithms** with the following restrictions:
   a. Does not store any prior tally, i.e. recalculate the frequency for every characters
   b. Use additional memory space to store the prior tally somehow.

You can give pseudo code for the algorithms. Compare the time **and space** complexities of the two approaches. Space complexity apply the same idea as time complexity but focus on memory usage.

5. [Git] Consider the following scenario, suggest how to achieve the desired outcome by utilizing Git.
   a. You are the working on a **solo C coding project**.
   b. There is one **function X** in the project that has two **possible implementations A and B (e.g. different algorithms, different data structure etc).**
   c. You want to **try both of the approaches separately**.

Focus only on functionalities learned in the studio. Discuss the problems with your approach.

**~~~ End of Tutorial ~~~**