CG1112 Engineering Principle and Practice II

Semester 2 2018/2019

Week of 28th January 2019

Tutorial 1 Suggested Solutions

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 used both a Raspberry Pi and Arduino. Find out the differences between these boards in the follow areas:
    a. Power Requirement (Voltage, Current for typical usage).
    b. Hardware Specification (CPU clock speed and runtime memory).
    c. Interfacing capabilities (to other devices, components, networking etc).
    d. Software environment

   Based on the above, suggest a 1-2 scenarios where Pi is more suitable for deployment.

   [For better consistency in discussion, please base your findings on **Raspberry PI 3 model B** and **Arduino Uno**.]

2. [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 this approach.

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.

**~~~ End of Tutorial (Part 1) ~~~**