Assignment 9

Jackson Fox

Due 12/2/2021

Task 1

a) Unit Testing

Unit testing is testing of the performance of specific functions or individual units of a software project. Unit testing focuses focuses on small specific chunks of code that individuals work on and then merge back into main branches and the overall project - these smaller chunks usually only deal with a small number of inputs and outputs.

b) Integration Testing

Integration testing is testing that evaluates the performance and behavior of a larger group of software modules working together. This testing would take the modules validated in many different unit tests and make sure that the parts are interacting as expected and that any hand off locations between units occurs as expected.

c) Functional Testing

Functional testing focuses on validating that a specific section of a project behaves as intended to an end user. Where unit testing focuses on specific small in and out puts for various specific tasks, functional testing is used to validate that the system is behaving as an end user would expect and need based upon the specifications of the project developed throughout the SDLC. Functional testers generally have no knowledge of how the software was actually coded, and focus exclusively on the actual behavior of the software.

d) End-to-end Testing

This testing is used to follow test an entire software workflow from start to finish. This can include processes such as the transmission of a packet of information from a system to a database to another system - End-to-end testing is used to validate that the task can be completed as intended from the trigger to the end of the defined process flow. This testing

usually includes tracking the transmission of a specific item of data and any subsequent artifacts that it generates from generation through to the final resultant process.

e) Performance Testing

Performance testing focuses on how the system behaves under varying workloads while executing its designed workflows. Specifically, it focuses on validating that the systems running the processes in question complete the task(s) within a set threshold of performance parameters, usually time to complete or overall memory used. Especially important with recurring batch tasks that need to be completed within certain windows.

f) UsabilityTesting

Usability testing is performed to confirm that not only does a software satisfy the design requirements (validated in acceptance testing), but that is also is intuitive, accessible and effective for the end users. This testing is to make sure that the implementation of solutions to the design requirements is effective in working with the intended audience, and the way that they complete their tasks. As there are many ways to satisfy requirements, usability testing helps identify if the right approach has been taken to optimize the interface of the software and the end user.

g) Acceptance Testing

Acceptance testing is completed by end users interacting with the system and validating that the behavior and performance is as expected based upon the functional specifications of the system. Acceptance testing formally validates that the software/project aligns with the goals of the end users and works within their process flows and continues to allow them to achieve the tasks that their position/job requires, and that it satisfies the documented and contracted requirements of the project. Acceptance testing is set apart from other testing as that the testers are the end users of the software, who often do not have any involvement in the actual development.

h) Beta Testing

Beta testing is similar to Acceptance and usability testing, but is distinct in that a nearly finalized product is released to a larger group of users than the specific usability and acceptance testers, in order to see if there are issues that may have not been captured in the testing scripts and formal approaches taken in the testing process. Beta testing usually involves a very large number of users, and is much less formal than other testing - the sandbox approach allows for identification of bugs that may not be otherwise found.

i) Additional Testing Types

During my time as a Technology Consultant, our teams utilized Quality Assurance Testing in addition to User Acceptance Testing - the test scripts that were executed were essentially the same but the main difference was that QAT was executed internally whereas UAT was completed by the client. Also, the QAT team had intimate knowledge of how the system was designed in order to test non-intuitive functional situations to ensure that they don't break the system based on how the system was built.

Task 2

a)

Complete

b)

Submitte

c)

There is an extremely exponential relationship in the amount of time needed as n increases. This occurs because every addition n+1 to the size of the rows/columns, the overall number of entries increases by a value of 2n+1. Additionally, using the multiplication function provided for this task, there are n^3 multiplications and n^3 additions that occur in multiplying A by B for a given row/column length n. This means that for every +1 addition to the size n (n+1), the number of operations increases by $6n^2+6n+2$. Hence, the exponential increase in time that we see with the increasing n. If you plot the results on a log/log scale, you see a linear plot with a slope of around 10, which implies that the time is related to N in a polynomial expression with a magnitude of 10.

d)

For reading data from registers: $24n^3$ bytes
Reading the data happens each time the function adds a value to the current C entry during the innermost for loop, which occurs n^3 times. For that process, a value from all 3 arrays must be read, and each double entry in those arrays is 8 bytes.

For writing data to main memory: $8n^2 + 8n^3$ bytes
Writing the data to main memory is only done for the C array, but is done twice. Once to give each entry 0.0 as a value, which happens n^2 times, and then once to update the value

of the C entry as the multiplication occurs, which happens n^3 times. There are 8 bytes per entry as the array is of doubles.

Task 3

a)

Complete

b)

The primary difference between the two functions is that in Test1, the volatile variable is not called every time in the actual loop, whereas it is in the Test2 function. However, because the loop is related to the volatile variable in Test1, the loop is not optimized away. The reason that the Test2 is consistently faster than the Test1 function as the step size decreases is due to the extra step of having to find the volatile variable and assign the value at the end of the loop, vs. in the Test2 where that step is left out. If you look at the timing differences for most of the loops, the difference between 1 and 2 is almost the same, indicating a similar operation difference for both.

c)

As stride increases, the time taken goes down for both of the testing functions. I would imagine that this occurs because there are simply less calls to be made to get data as we jump through the loop in larger and larger steps. I imagine that the reason the timing decrease begins to taper off as the stride increases past 8 is because there are diminishing returns in the number of calls that are being executed as the step size increases.

d)

Pass

Task 4

a)

Complete

b)

Complete

c)

The second swap method is faster, where the swap goes column by column. The timing difference is due to the fact that R stores arrays in column major in memory (as does MATLAB and python)- therefore the items in the same column will be more localized in memory requiring less time to find for each iteration of the swap.