### 

### Computing, Digital Forensics and Cyber Security

#### BSc Computer Science 2019-2020

### Mathematics for Computer Science

### Assignment 2

**Date: 10/12/2019**

**Mathematical Algorithms**

General guidelines for submission

1. This is an **individual assignment** and must represent your own work.
2. The required date of submission is **9th Jan 2019 14:00** via Blackboard as a zip folder called **MCOMD1MC.**
3. The report must adhere to the department guidelines.
4. This assignment has been set by **James Cugley**
5. This assignment has been moderated by **Alexios Louridas**
6. This assignment is worth 50% of the Maths for Computing module.

General advice

1. You are required to back up your work regularly onto your drive, and onto removable storage devices of your choice. Always check the date-stamp on your files before submission. You will need to submit the latest version of your software.
2. You submission must be compatible with software versions we currently have on the University’s network.

Tested Learning Outcomes

1. Demonstrate knowledge of the mathematics used in computing;
2. Demonstrate and apply the concept of mathematical abstraction to problems;
3. Understand how to translate mathematical abstraction to computers;
4. Understand the benefits of different computational implementations.

Introduction

The aim of this assignment is to utilise three or more subject areas within mathematics to devise algorithms to solve problems in the fields and linear algebra and game theory. The assignment requires a good understanding of matrix manipulation and probability theory as well as other standard techniques taught earlier in the module. Some independent research will be necessary in order to fully understand each field and receive full credit.

Task 1

If z1=9+j7 and z2=6-j3, perform the following operations:

Once you have solved the problem manually and given documented proof of each step taken, write the necessary lines of code in Python to solve the same problem using a program.

Task 2

Use Cramer’s rule to solve the series of linear equations given in the example below:

Again, once you have solved the problem manually and given documented proof of each step taken, write the necessary lines of code in Python to solve the same problem using a program.

Briefly discuss any complications that might arise when trying to generalise the algorithm so that it solves a different series of linear equations and suggest possible applications to STEM that might arise from sophisticated generic solutions that include multiple variables and equations.

Task 3

You play a game where you bet some money and have a probability p to win twice the money (i.e. get back your money then the same again – if you bet one pound, you get back two pounds making you one pound richer), otherwise you lose your money. You have a strategy to beat the game:

• You first bet one pound. If you win, you are one pound up and you stop.

• If you lose, you then bet two pounds. If you win, you get four pounds back, which minus the two pounds you bet on this game and the one pound you lost on the previous game leaves you up by one pound. You stop in this case.

• If you lose that one, you then bet four pounds. Winning this will therefore leave you one pound up overall – in which case you stop.

• If the previous one was lost, you then bet eight pounds – and so on, doubling your bet each time.

It would seem therefore that you have beaten the system – you will always stop one pound up. The problem with this approach is that you have a finite amount of money you can bet, and if you lose too many times in a row, then you don’t have money to carry on going.

Write a code to simulate this strategy. Your code should have as an input the probability p of winning each game; and the total amount of money you have to start with, x. The code should work out what the probability of winning (i.e. ending up one pound richer) is, as well as the average amount you win (which isn’t completely simple as if you win, you get one pound but if you lose, then you lose everything you have paid in). To visualise the results, suppose your maximum balance is 2n − 1 (you will find this allows you to play a maximum of n rounds before you are bankrupt). Plot a graph of the probability of winning as a function of n, as well as the expected (average) winnings as a function of n for the case where it is a fair game p = 0.5, and for the case where the house has an advantage p = 0.4. Choose a sensible maximum value of n to plot your graph until, and comment on your results.

NB: Any Python coding is not included in the word count.

Deliverables

* Professionally written and compiled report for each task covering the key aspects of the design and implementation of the algorithms as well a written commentary on all research elements within the assignment brief.
* Documentation on any worked solutions or proofs outlined in the assignment brief were appropriate.
* Appendices containing references to any code sourced from elsewhere and used within the implementation and any high level utilities or libraries available in Python;
* Source and possible compiled files of the implementation in a suitable file structure for testing within the standard Python environment.

BSc Computer Science Year 1

2019-2020

Maths for Computing

### Assignment 2

**Date Set: 10th December 2019**

**Course Tutor: James Cugley**

**Feedback Sheet**

|  |  |  |
| --- | --- | --- |
| **Student’s Name:** | | **%** |
|  | **Max Mark** | **Awarded** |
| **Report:** | **10** |  |
| Professionalism | 5 |  |
| Technical Exposition | 5 |  |
| **Task 2:** | **30** |  |
| Matrix Based Algorithm | 15 |  |
| Proof | 5 |  |
| Discussion | 10 |  |
| **Task 1:** | **30** |  |
| Matrix Based Algorithm | 15 |  |
| Proof | 5 |  |
| Discussion | 10 |  |
| **Task 3:** | **30** |  |
| Probability Based Algorithm | 15 |  |
| Proof | 5 |  |
| Discussion | 10 |  |
| **Total:** | **100** |  |
| Comments: |  |  |

Marking scheme

Each task is graded out of a maximum of 30 marks and banded according to the following criteria:

|  |  |
| --- | --- |
| **Banding:** | **Description:** |
| 0-11 marks (fail) | Missing answer and/or code and very weak response to task. All written commentary lacks coherence and all theoretical elements are missing. Little or no referencing. Answers, if any, not relevant to task with inappropriate coding or irrelevant examples of coding. |
| 12-14 (below average pass) | Relevant response to task, but missing elements and only partially working code where required. No proofs evident. Overall, there is a weak account of theoretical elements as required. Some evidence of referencing. Tendency to be very descriptive. Does not demonstrate a mastery of PYTHON. |
| 15-17 (pass) | Reasonable response to task, but with some missing elements with some code weakness apparent. Proofs provided, but not necessarily always appropriate. Reasonable theoretical account with references included. Mostly descriptive, but some analytical content included in responses. Little knowledge of STEM. Some understanding of PYTHON. |
| 18-20 (good pass) | Good response to task with all elements covered although may show some weaknesses in coding. Well referenced commentary with tendency to be more analytical rather than purely descriptive. Proofs provided and relevant, but my contain some truncation. Limited evidence of working beyond the curriculum, but with a partial understanding of relevance to STEM. Good use of PYTHON including libraries where necessary. |
| 21+ (excellent pass) | Concise responses to task with efficient well-documented code where appropriate showing mastery of development environment. Well-argued discussions thoroughly referenced throughout. All proofs well worked and entirely relevant to the task. All discussions analytical in nature with concise descriptive accounts given where necessary. Clear evidence of research beyond the module curriculum and appropriately applied to the task. Complete understanding of task and its relevance to STEM. |