MSc in Bioinformatics for Health Sciences

APA. Advanced programming, algorithms and data structures

Syllabus Information

Academic Course: 2019/20

Academic Center: 804 - Official Postgraduate Programme in Biomedicine

Study: 8045 – Bioinformatics for Health Sciences - MSc

Subject: 32546 - APA. Advanced programming, algorithms and data structures

Credits: 5.0

Course: 1st

Teaching languages: English

Teachers: Emre Guney

Teaching Period: 1st term

Presentation

This course will cover advanced concepts in algorithm design, data structure and programming. The course will focus on building the capacity to improve the computational and space efficiency of algorithms and the ability to translate these algorithms to efficient implementations. Key concepts that are going to be introduced in the course are (i) [Algorithms] complexity theory, divide-and-conquer scheme, dynamic programming, greedy search, approximation algorithms; (ii) [Data structures] stacks, heaps, suffix trees and graphs (iii) [Programming] call by value vs reference, static vs dynamic typing, recursion, object-oriented programming, polymorphism and generic programming. The course will involve several programming exercises, a written exam and a final project to apply these advanced concepts in a practical and real-world setting.

The course will build upon the fundamental algorithms, data structure and programming knowledge (i.e. Bioinformatics undergrad course ALG 1, MSc course ALG or equivalent CS/engineering coursework). Good knowledge of Python (or an equivalent high level language such as C++, Java) is required. Depending on the need, for those who are not familiar with concepts such as computational complexity, data structures, recursion and object-oriented programming, a brief overview will be provided during the course.

Associated skills

General competences:

1. Gaining confidence in programming, improving auto-sufficiency in solving programmatic tasks and working in a team setting.
2. Learning advanced concepts of complexity theory and algorithm design.
3. Understanding fundamental concepts in high level programming languages.
4. Acquiring skills required to effectively create and use advanced data structures.

Specific competences:

1. Building capacity to distinguish P / NP problems via computational complexity analysis.
2. Acquiring proficiency at programming using advanced concepts such as object oriented programming.
3. Introduction to the fundamental programming language concepts such as call by value vs reference, static vs dynamic typing, generic programming (e.g., using typing library in Python 3 or templates in C++).
4. Understanding the differences across various searching, sorting and graph traversal algorithms and the data structures used for implementing them.
5. Familiarizing with different randomized and approximation algorithms in Bioinformatics such as Max-Cut and Vertex Cover as well as various data management and processing strategies in Bioinformatics such as phylogenic trees, data parsing and serialization using XML and JSON.
6. Practical implementation of efficient scalable and reusable code.

Contents

Block 1: Computational Complexity and Design of Algorithms

1.1. Introduction to complexity theory

1.2. Searching and sorting algorithms: Divide-and-conquer scheme, greedy search, binary search, hashing, bublesort, mergesort, quicksort

1.3. Graph algorithms: Dynamic programming, breath-first search, depth-first search and minimum spanning trees

1.4. Randomized and approximation algorithms in Bioinformatics

Block 2: Advanced Data Structures

2.1. Lists and stacks

2.2. Heaps and queues

2.3. Trees and graphs

2.4. Data management and processing in Bioinformatics

Block 3: High-level Programming Language Concepts

3.1. Values and parameters: Call by value vs reference

3.2. Types: Static vs dynamic typing

3.3. Recursion and exception handling

3.4. Object-oriented programming: Interfaces and inheritance

3.5. Polymorphism and generic programming: Operator overloading and templates

3.6. Reflection and native function calls

Block 4: Recapitulation and open discussion

4.1. Code optimization

4.2. Summary

4.3. Discussion

Teaching methods

The course will be focused on introducing the fundamental concepts and skills for improved understanding of advanced computation through the use of high-level programming languages. The classroom lectures aim to provide various techniques and data structures widely used in computer science and to establish the competence for making algorithmic and data structure related decisions considering the underlying computational complexity. The practical coursework elements support these objectives and helps expanding the capacity to understand the role of various data structures algorithms and heuristics in Bioinformatics and Computational Biology.

Training activities

1. The fundamental concepts pertinent to theory of computation, data structures and high-level programming languages will be discussed in the classroom.

2. A key component of the class is the practical sessions during which the participants will be working on the implementation of concepts introduced in the lectures. Some of the tasks in these sessions will require continued outside classroom activity for their completion.

3. The participants will deliver a brief report (1-2 pages), describing their solutions to algorithmic challenges and programming assignments explained in the classroom or during practical sessions.

4. A final project will be developed to apply the contents of the course in a practical and real-word setting. The projects will be assigned at the beginning of the course and are going to developed progressively during the course.

Evaluation

The students will be evaluated according to their performance in practical programming tasks, written exam and a project in which a small group of students will work together. The practical task performance evaluation will be based on the interest and ability to solve the practical challenges. The written exam will include problems on the theory and the topics learned in the class. The projects will be presented to the rest of the class at the end of the term and all the students will participate in their evaluation.

Grading system:

Practical session performance & assignment evaluation (20%).

Written exam (30%).

Project (50%).

A minimum performance of 50% on each item is required to pass the subject.