Course Unit Descriptor

**Study Programme:** Information Technologies

**Course Unit Title:** Data Structures and Algorithms 2

**Course Unit Code:** IT202

**Name of Lecturer(s):** Miloš Savić

**Type and Level of Studies:** Bachelor Academic Degree

**Course Status (compulsory/elective):** Compulsory

**Semester (winter/summer):** Winter

**Language of instruction:** Serbian (primary), English (secondary)

**Mode of course unit delivery (face-to-face/distance learning):** Face-to-face

**Number of ECTS Allocated:** 7

**Prerequisites:** None

**Course Aims:**

The objective of the course is to introduce students to fundamental sorting and searching algorithms, related abstract data types and general algorithmic techniques.

**Learning Outcomes:**

*Minimum:* Successful students should be capable to use the implementation of presented algorithms and abstract data types to solve practical programming problems.

*Desirable:* At the end of the course it is expected that successful students deeply understand and are able to implement presented algorithms, techniques and abstract data types, and apply them in a variety of practical programming problems.

**Syllabus:**

*Theory*

Elementary sorting algorithms. The priority queue abstract data type (ADT). Advanced algorithms for sorting arrays and lists. Hash functions. The set ADT and the map ADT realized by open and closed hash tables. Divide and conquer algorithms. The backtracking algorithmic technique and its applications to combinatorial enumeration and optimization problems. Greedy algorithms. Branch and bound algorithms. Dynamic programming. General and binary trees. Algorithms for searching and traversing binary trees. Prefix codes and Huffman coding trees. The set ADT and the map ADT realized by binary search trees. String distances. Elementary and advanced string searching algorithms.

*Practice*

Typical applications of the presented algorithms, algorithmic techniques and abstract data types. Individual practical programming tasks related to sorting, hash functions and tables, backtracking, dynamic programming, and tree searching and traversal algorithms.

**Required Reading:**

1. Robert Sedgewick and Kevin Wayne. Algorithms, Fourth edition. Addison-Wesley, 2011.
2. Michael T. Goodrich, Roberto Tamassia and Michael H. Goldwasser. Data structures & algorithms in Java, Sixth edition. Wiley, 2014.
3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to algorithms, Third edition. The MIT press, 2009.

| | | |
|---|---|---|
| **Weekly Contact Hours:** 5 | **Lectures:** 2 | **Practical work:** 3 |

**Teaching Methods:**

Theoretical classes are based on the classical teaching model involving a projector. Presented algorithms, techniques and data types are augmented with illustrative case studies implemented Java (e.g. the n-queens problem, the knapsack problem, string edit distance, closest pair of points, etc.). At theoretical exercises, solutions of practical programming problems are presented and discussed with students. At practical exercises organized in computer labs, students have to individually solve practical programming problems involving algorithms, techniques and data types covered by theoretical classes. To approach the oral exam students have to pass pre-exam obligations consisting of 4 practical programming tasks conducted on computers. At the oral exam students are expected to demonstrate the in-depth understanding of the topics covered by the course.

**Knowledge Assessment (maximum of 100 points):**

| **Pre-exam obligations** | points | **Final exam** | points |
|---|---|---|---|
| 4 practical tasks | 50 (10 + 10 + 15 + 15) | Oral exam | 50 |

The methods of knowledge assessment may differ; the table presents only some of the options: written exam, oral exam, project presentation, seminars, etc.