Universidad Zaragoza

Year : 2020/21

30232 - Algorithms for Difficult Problems

Syllabus Information

Academic Year: 2020/21
Subject: 30232 - Algorithms for Difficult Problems
Faculty / School: 110 - Escuela de Ingeniería y Arquitectura
Degree: 439 - Bachelor's Degree in Informatics Engineering
ECTS: 6.0
Year: 4
Semester: First semester
Subject Type: ---
Module: ---

1.General information

1.1.Aims of the course

The course and its expected results meet the following approaches and objectives:

In this course the students will improve their ability to design and develop algorithms focusing on probabilistic, approximate and heuristic techniques. Students will learn to recognize problems for which the solution uses these techniques, to design algorithms that use them, and to justify the correctness and efficiency of those algorithms.

1.2.Context and importance of this course in the degree

The Computer Science specialization covers a wide range of concepts, from the theoretical and algorithmic foundations to the forefront of developments in bioinformatics, robotics, computer vision, video games, and other interesting areas. Algorithms for difficult problems is the second of the courses of the specialization and is intended to complement the course of Basic algorithms with popular probabilistic algorithms and use of heuristics, among other.

1.3.Recommendations to take this course

Interest and effort are required, in addition to the knowledge acquired in previous couses of mathematics, programming, data structures and algorithms, computer theory, and basic algorithms.

2.Learning goals

2.1.Competences

After passing the course, students will be more competent to ...

1. Combine general knowledge and specialized engineering to generate innovative and competitive proposals for professional activity.
2. Solve problems and make decisions with initiative, creativity and critical thinking.
3. Communicate and transmit knowledge, skills and abilities in Castilian and English.
4. Use the techniques, skills and tools necessary for engineering practice thereof
5. Learn continuously and develop independent learning strategies
6. Apply the information and communications technology in Engineering
7. Have a thorough understanding of fundamental principles and models of computation and know how to apply them to interpret, select, assess, model, and create new concepts, theories, uses and technological developments related to computer science.
8. Assess the computational complexity of a problem, knowing algorithmic strategies that can lead to resolution and recommend, develop and implement one that guarantees the best performance according to the requirements.
9. Know the fundamentals, paradigms and own techniques of intelligent systems and analyze, design and build systems, services and applications that use these techniques in any scope.

2.2.Learning goals

The student, after passing this course, achieves the following results ...

1. He/she knows the inexact computing models and is able to select the right one and use it for the modeling of heterogeneous application domains.
2. He/she is able to identify NP-hard problems.

3. He/she knows probabilistic, approximate and heuristic algorithmic techniques for NP-hard problems.
4. He/she can identify the most relevant components of a problem and select the most appropriate approximate technique to solve it, as well as give the reasons for his/her choice.
5. He/she knows how to compare problems and use this comparison to solve a problem from an efficient solution of another one.
6. He/she knows how to reason about the error rate and efficiency of approximate and probabilistic algorithms.
7. He/she knows how to apply analysis of amortized efficiency to advanced data structures.
8. He/she has the ability to work in a group, identify group goals, map out a work plan to achieve them, recognize the different roles within a team and is committed to the assigned tasks.
9. He/she can manage independent learning and development including management and organization time.
10. He/she appreciates the need for lifelong learning.

2.3.Importance of learning goals

Difficult problems from the computing poitn of view are usually identified with NP-hard problems that are studied in this course and cover interesting problems in all areas of knowledge. These are problems for which the usual algorithmic strategies are very inefficient, or even fail to provide the desired solution. The inexact algorithmic techniques are suitable for many of these problems and allow us to reflect useful insights for building algorithms.

3.Assessment (1st and 2nd call)

3.1.Assessment tasks (description of tasks, marking system and assessment criteria)

The student must demonstrate that he has achieved the intended learning outcomes through the following evaluation activities

SUMMARY:

recommended option (progressive release from final exams):

Practical part:
Laboratory Practice (Group) during the semester: 20%.
Part of theory and problems:
Individual exercises during the semester: 20%.
intermediate written test: 30%.
Final exam (performed only in part): 30%.

Option based solely on final exams:

Practical part:
(Individual) ~~laboratory~~ programming test: 20%.
Part of theory and problems:
Final exam (in full): 80%.

DETAILED EXPLANATION:

recommended option (progressive release of final exams):

- Throughout the semester practical programming work will be proposed and must be solved in the laboratory. For this purpose teams will be formed by a certain number of students to be fixed at the beginning of the course. The papers presented by the students will be graded with a quantitative rating from 0 to 10. To obtain those grades programs will be assessed according to specifications, the quality of its design and presentation, proper application of the methods of resolution, time spent, and the ability of each team member to explain and justify the design done. Students who have met the deadlines set for the programming lab work, and have demonstrated in them a level of achievement and quality of adequate results, will obtain a grade in the practical laboratory programming part, weighing 20% of the final grade for the course.
- In addition, throughout the semester homework sheets for individual work will be published, from which students may submit, no later than the date specified in each sheet, a certain number of exercise solutions to be set at the beginning of the course. The exercises presented by the students will be graded with a quantitative grade of 0 to 10. IMPORTANT NOTE: Before submitting an exercise solution among those proposed in a sheet the student must inform the teacher, selecting the exercise in moodle or as otherwise indicated, and wait for that teacher authorizing the carrying out of the exercise. Students who have met the deadlines set for these homework exercises will get the corresponding "individual work exercises" mark, weighing 20% ??of the final grade for the course.
- At about half semester there will be an intermediate written test consisting of individual solution for 50 minutes of an exercise proposed by the teacher. The intermediate written test will be graded with a quantitative rating from 0 to 10. Students who have completed the intermediate written test, will obtain the grade "intermediate written test", weighing 30% ??of the final grade for the course.
- In the written exam the student must solve problems similar to the weekly exercises proposed during the semester and to the problems in the intermediate written test, he must also answer conceptual questions and provide solution to exercises that prove his achievement of the learning results required in the course. Students who choose the "progressive release of final exams" option will be exempt from part of the written exam and must take only a

mandatory part of the exam, to be determined at the same time. The grade obtained in this mandatory part of the exam weighs 30% of the final grade for the course.

Option based solely on final exams (global evaluation):

The global evaluation of the course consists of two parts:

- Individual programming test. In each call a practical laboratory programming test will be given, which will propose the student exercises similar to those made in the lab classes or in class. The grade obtained weighs 20% of the final grade for the course.
- In the written exam the student must solve problems similar to the weekly exercises proposed during the semester and to the problems in the intermediate written test, he must also answer conceptual questions and provide solution to exercises that prove his achievement of the learning results required in the course. Students who choose the "Option based solely on final exams" should take the written exam in full. The grade obtained in this written exam in full weighs 80% of the final grade for the course.

4.Methodology, learning tasks, syllabus and resources

4.1.Methodological overview

The methodology followed in this course is oriented towards the achievement of the learning objectives. A wide range of teaching and learning tasks are implemented such as:

- Learning concepts and methodologies for the design and implementation of correct and efficient algorithms through lectures, in which student participation is encouraged.
- The application of such knowledge to the design and analysis of algorithms and programs in the classes of problems. In these classes, students will play an active role in the discussion and resolution of problems.
- Teamwork developed to address the lab practical work of the course, the result is reflected in the delivery of suitably designed and documented programs, as well as the explanation and justification of the design made and decisions taken.
- Continued work on the understanding that combines concepts, analysis and programming problems solving using "pencil and paper" and the set-up in some computer programming projects.
- The study and continued work from the first day of class.

4.2.Learning tasks

The course includes the following learning tasks:

- 1 Classes taught in the classroom will develop the syllabus of the course.
- 2 In the classes of problems the concepts and techniques presented in the course syllabus will be applied. Problems and exercises will be proposed beforehand and different solutions to these problems will be presented and discussed in class. Other exercises will also be proposed during the problems sessions to be solved during it, some of them individually and others to be worked in groups.
- 3 Practical work takes place in a computer lab or personal computers of students at home. In these sessions, students will work in teams and perform a number of programming assignments directly related to the topics studied in the course. For this, a series of tasks or programming exercises will be proposed for the students to solve in groups and delivered within the time limits fixed in each case.

4.3.Syllabus

The course will address the following topics:

- 1. Introduction. The NP-hard problems. Optimization problems.
- 2. Approximate algorithms. Concept. Algorithm design. Guarantees and limits.
- 3. Probabilistic Algorithms: Las Vegas and Monte Carlo. Analysis. Pseudorandom generators.
- 4. Heuristics. Simulated annealing (simulated annealing).
- 5. Genetic algorithms.
- 6. Amortized analysis. Advanced data structures.
- 7. Specialized algorithms.

4.4.Course planning and calendar

Calendar of sessions and homework presentation

The general organization of this course is as follows.

? Theoretical classes (2 hours per week)

? Classes of problems (1 hour weekly)

Presentation of exercises and practical work:

The exercises proposed in class and lab work scheduled to be done in the laboratory or at home should be performed and presented when specified in each of them, and within deadlines to be announced in the description of each of the proposed works or well in advance.

Student Work

The time needed by the student to achieve the learning outcomes in this course is estimated in 156 hours distributed as follows:

? 45 hours, approximately, of classroom activities (theoretical and problem-solving classes);

? 45 hours of team programming work to develop proposed laboratory work programs;

? 60 hours of effective personal study (study notes and texts, problem-solving, class preparation, program development);

? 6 hours of a written final theory exam and practical examination in the laboratory.

The exam date and deadlines for homework and lab assignments will be announced well in advance on the website of the course.

4.5.Bibliography and recommended resources

[BB: Bibliografía básica / BC: Bibliografía complementaria]

- [BB] Brassard, Gilles : Fundamentos de algoritmia / G. Brassard, T. Bratley ; traducción, Rafael García-Bermejo ; revisión técnica, Narciso Martí, Ricardo Peña, Luis Joyanes Aguilar . - 1ª ed. en español, reimp. Madrid [etc.] : Prentice Hall, 2008
- [BB] Hromkovic, Juraj : Algorithmics for hard problems : introduction to combinatorial optimization, randomization, approximation, and heuristics / Juraj Hromkovic . - 2nd ed., corr. Berlin [etc.] : Springer-Verlag, 2004
- [BB] Introduction to algorithms / Thomas H. Cormen ... [et al.] . - 3rd ed. Cambridge, Massachusetts ; London : MIT Press, cop. 2009
- [BB] Skiena, Steven S. The algorithm design manual / Steven S. Skiena . 2nd ed., corr. London : Springer, cop. 2012
- [BC] Dasgupta, Sanjoy. Algorithms / Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani . Boston [etc.] : McGraw Hill Higher Education, cop. 2008
- [BC] Kleinberg, Jon. Algorithm design / Jon Kleinberg, Éva Tardos . Boston : Pearson/Addison-Wesley, cop. 2006
- [BC] Vazirani, Vijay V. : Approximation algorithms / Vijay V. Vazirani . - [2ª ed.], reimp. Berlin : Springer, cop. 2010