Algorithms (Spring 2015)

Brief course description
This masters level algorithms course aims to refine the algorithmic thinking of a student, moving her/him from an algorithm consumer to an algorithm designer. The project-oriented course structure is tailored to facilitate the transition. The first part of the course focuses on the analysis of algorithm complexity and correctness, amortised and probabilistic analysis, while we later discuss advanced data structures, approximation algorithms, heuristic methods and biologically-inspired computing. The course also introduces computational geometry, and parallel and distributed computing.

Expected level of expertise
- **Undergraduate level algorithms knowledge.** This can be acquired through Algorithms and Data Structure 1 ([http://www.fri.uni-lj.si/en/education/10091/class.html](http://www.fri.uni-lj.si/en/education/10091/class.html)) and Algorithms and Data Structure 2 ([http://www.fri.uni-lj.si/en/education/10029/class.html](http://www.fri.uni-lj.si/en/education/10029/class.html)) courses taught as undergraduate courses at FRI, or through similar courses available elsewhere.
- **Solid programming proficiency.** The class project will include a lot of programming! You are free to choose any programming language, but some tasks/topics are more suited for a certain language. Python is probably the most widely applicable across all topics, followed by Java. Certain projects require a basic knowledge of database access.

Instructor
name: Veljko Pejović
email: Veljko.Pejovic@fri.uni-lj.si (put “Algorithms 63508” in the subject line; please use the course webpage for any questions that may be of general interest for people in the class)

Teaching assistant
name: Erik Štrumbelj
email: erik.strumbelj@fri.uni-lj.si

Course meetings

<table>
<thead>
<tr>
<th></th>
<th>Lectures</th>
<th>Lab sessions</th>
<th>Office hours Veljko</th>
<th>Office hours Erik</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>T 2pm-5pm</td>
<td>TBD</td>
<td>W 12:00-14:00</td>
<td>F 13:00</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>P18</td>
<td>TBD</td>
<td>Rm 3.15</td>
<td>TBD</td>
</tr>
</tbody>
</table>

Resources

<table>
<thead>
<tr>
<th>Course website</th>
<th><a href="https://ucilnica.fri.uni-lj.si/course/view.php?id=315">https://ucilnica.fri.uni-lj.si/course/view.php?id=315</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTE:</td>
<td>The course website is a main point for all the time-sensitive course information, for all class-related discussions, lecture and lab session materials, and for additional assigned readings. It is your responsibility to ensure that you are following updates from ucilnica.fri.uni-lj.si.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Textbooks</th>
<th>The following books are listed in the order of importance.</th>
</tr>
</thead>
</table>
Full course description

Undergraduate computer science courses equip students with the basic toolbox of algorithms and data structures, and with the ability to think algorithmically -- to pose a problem and model a solution so that it can be efficiently run on a computer. While this may be enough for many practical applications, certain problems require either sophisticated tools that are outside of the basic undergraduate knowledge, or are so complex that there are no known accurate, yet efficient solutions available. Sometimes, the change of the thinking paradigm, and casting a problem to, for example, geometry, or getting inspiration from biology can increase our chances. Furthermore, there are problems for which using more than a single processor or a single machine is the best way to proceed.

This course starts with the analysis of the algorithm time and space complexity, with respect to the input provided. The analysis primarily focuses on algorithms that are less straightforward to analyse -- recursive algorithms. The master theorem and the Akra-Bazzi method are the key approaches we use for this purpose. Further, we present amortised analysis, where we average over all the instructions performed by an algorithm in order to discard the effect of a single “misbehaving” instruction. We then show how randomisation of the input can, in some cases, help with the expected algorithm complexity, and discuss randomised analysis.

The course then moves to advanced aspects of topics students are already familiar with, such as sorting and searching. For example, we will see that sorting can be done faster if direct comparison among items avoided. We also introduce new structures such as van Emde Boas trees, Kd-trees, range trees, interval and segment trees. These structures greatly simplify certain tasks, such as searching in space.

Computational intractability is a property of a problem indicating that no efficient (polynomial time) solution for the problem can be constructed. Thus, we aim for the next best thing -- a solution that is guaranteed to be close to optimal. Approximation algorithms can get us there, and can tackle difficult problems such as the traveling salesperson and the vertex cover problem, to name a few. Afterwards, we concentrate on algorithms called heuristic or local search algorithms, which explore the space of possible solutions in a sequential manner. It is often hard to prove that these algorithms arrive to a “good” solution, yet they are immensely important for many practical applications: Metropolis-Hastings algorithm has been named among the top ten algorithms of the 20th century by the Society for Industrial and Applied Mathematics. Some of these algorithms are directly inspired by nature.

Computational geometry brings us back to the old days when people used to think about problems by using pegs, ropes and drawings in sand. However, stating problems in terms of geometry can be very

---

1 Cormen et al. is highly suggested as probably the most commonly used algorithms book in the world. Even if you don’t plan an academic career, get this book -- many interview questions for industry jobs come exactly from this book.

2 The two relevant chapters from this book are posted on the course webpage.

3 The two relevant chapters from this book are posted on the course webpage.

4 You can download a pdf of this book for free from the University network.
helpful, and is routinely applied in computer vision, robotics, geographic information systems, and many other purposes. From this vast field, we concentrate on identifying relationships among lines and points in 2-D space, finding convex hulls (wrappers around a set of points), and guarding areas with triangulation.

We then discuss the opportunities to combine multiple processors, or even multiple machines residing in different locations, to speed up computation. Smartphones use multithreaded computing, which balances simultaneous execution of multiple tasks, all the time, so that you continue to interact with the touch screen even while an unrelated computation is happening in the background. But the true power of parallelism comes with multiple processing cores on a CPU. Not all algorithms can be parallelised, and there is a limit, expressed by the Amdahl's law, on how much benefit we can get by increasing the number of processing units. Distributed computation has become increasingly important since the advent of the Internet. Here, we often do not have a tight centralised control over multiple processors. Rather, we have a set of machines that exchange messages over a communication network. We investigate how basic tasks such as sorting and searching are done over a distributed system.

Finally, the core of the course is a practical project that exposes a student to a real-world situation in which advanced algorithms need to be applied. Chances are that in such a situation solutions from textbooks cannot be directly translated to reality. Instead, one needs to be creative, strategically explore different options, and experiment with practical implementations, until a satisfactory solution is found. Advanced algorithms is such a vast field that in practical projects students may, and are indeed encouraged to, go beyond what has been covered in lectures.

Course goals
After you successfully complete the Algorithms course you will be able to:

- Analyse the complexity of a sequential or a recursive algorithm, describe the expected and the worst-case behaviour.
- Provide state-of-the-art solutions to sorting and searching problems using advanced data structures.
- Judiciously decide when an exact solution to a problem is not feasible, and use approximation and heuristic approaches instead.
- Solve problems in 2-D space using computational geometry algorithms.
- Improve algorithms through parallelisation and assess the improvement.
- Select an appropriate algorithmic approach for solving a complex real-world problem, extend off-the-shelf algorithms, and implement them in a functioning system.

Tentative course outline

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.2 - 27.2</td>
<td>Introduction, Course projects, Complexity revised</td>
<td></td>
</tr>
</tbody>
</table>
| 2.3. - 6.3. | Amortised analysis, Master theorem, Akra - Bazzi method              | Project proposal  
v1 due 2.3.  
v2 due 8.3. |
| 9.3. - 13.3. | Randomised algorithms, Probabilistic analysis                        |                                 |
| 16.3. - 20.3 | Sorting with assumptions: radix sort, bucket sort                    |                                 |
| 23.3. - 27.3. | Searching with assumptions: Van Emde Boas trees,                     |                                 |
### Course components

<table>
<thead>
<tr>
<th>Lectures</th>
<th>Lectures are essential to get the big picture of what you are learning about in this course and why. In the lectures you will be presented clear explanations of many algorithmic concepts that can be very hard to understand on your own. Moreover, not everything that is covered in the lectures comes from the textbooks. You will be able to ask for clarifications and occasionally express your opinion on how the course is progressing, thus directly influence the amount of learning that happens in the class. The attendance is mandatory (more about that in the course policies).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework</td>
<td>Homework assignments serve to help reinforce what you have learned during the lectures. They are tightly connected to the material covered in the lectures. You will be given two assignments throughout the semester. Assignments have to be submitted before their respective deadlines (see course website for details). You may not collaborate with others (not even your project partners) on homework assignments.</td>
</tr>
<tr>
<td>Lab Sessions</td>
<td>Lab sessions will mostly be dedicated to project work. Students will report on their progress on a weekly basis and discuss potential problems and challenges with the TA. You will get most out of these sessions if you prepare project-related questions for the TA beforehand.</td>
</tr>
<tr>
<td>Course project</td>
<td>The best way to understand a computer science concept is to implement it and test it. The course project lasts for the whole semester and requires you to 1) select a problem for which a solution is likely to be found within the course topics, and come up with a plan of building such a solution, 2) investigate the related work in the selected field, choose your approach towards solving the problem, and prototype one or more candidate solutions for the problem, or develop key pieces leading towards the solution, and finally 3) evaluate your solution(s) and share your findings.</td>
</tr>
</tbody>
</table>
from the problem formulation, investigation, solution development, and evaluation to a wider research audience. You can pick a project from the list of proposed problems/topics (on the course webpage), or you can come up with your own problem/topic. In case you propose your own project, it has to be related to the topic of advanced algorithms, it must not be a copy or a minor variation of a known solution, and it has to entail enough practical work to keep your team busy for a semester. Each project is done in teams of two or three people.

There are three milestones for the course projects.

The first one is the project proposal. Once you decide on your project, you will be asked to write a one page project proposal that should clearly state:

- the problem you are solving (along w/ background and related work)
- motivations (if this is your own idea) and challenges; why is this problem important and difficult?
- your proposed solution or approach and why it's new,
- your plan of attack with milestones and dates, and
- any resources you might need so we can take care of this early on in the semester.

The proposal should be 1-2 pages. The mark you get on the proposal will be a part of your overall project grade. Project proposals are submitted via the course website. Only one person per project team should submit, but the proposal should indicate full names of all other project team members.

Note that there will be two versions of the proposal. Proposal v1 will be due on the evening of Monday 2.3.2015. I will provide written feedback on the project proposals via email. Feel free to drop by during office hours to discuss and develop your project ideas further. Project v2 is the revised, more detailed and more thought-out version, to be submitted later the same week, on the evening of Sunday 8.3.2015.

The second milestone is the mid-semester presentation. Each project team will give a mid-semester progress presentation to the entire class on Tuesday 21.4.2015. Each presentation will give the audience a quick idea of the project motivations, approach to solving a problem, and current progress made by the team. The total presentation will be roughly 10 minutes (this number might change depending on the number of final project teams formed).

The third milestone includes the final project report and the final presentation. The final project report should not exceed 6 two-column pages using 10pt fonts. The content should be similar to a research workshop publication. Make sure you include enough detail for a reader to understand all of your design and experimental evaluation decisions. Your final report is evaluated according to the same standards that one would review a paper submission for a top workshop. The report is due on Friday 5.6.2015. 23:59. The presentations will be in class on Tuesday 9.6.2015., and should show the progress made since the mid-semester presentation, and the conclusions of your project.

The smoothest way to succeed in your project is to have a solid, realistic plan of work early on, to prepare an alternative in case your initial idea fails, to meet your TA and instructor frequently and talk about issues that prevent your work from progressing, and to balance labor across all the team members.

| Exam     | You will have a written final exam in the end. The exam questions will be very |
related to what has been taught in class and lab sessions. Note that relying solely on the material from the books might not be enough for you to successfully prepare for the exam. You will be given a practice exam near to the end of the semester. The exam is closed book -- no textbooks, or notes of any kind are allowed.

**Quizzes**
Occasional quizzes throughout the semester will provide you with early feedback about your strong and weak points and will help you prepare for the final exam. These quizzes will be posted online, will not be marked, but are mandatory.

**Readings**
Each lecture will have reading materials assigned to it. This will be either from the books or auxiliary materials posted on the course webpage. Reading the materials **before the class** will greatly help with understanding the topics taught.

**Marking**

**Course points are distributed as follows:**

- 50% Coursework, out of which:
  - 10% Homework assignments (there will be two assignments - 5% each);
  - 90% Class project, out of which:
    - 15% Project proposal
    - 35% Mid-semester presentation
    - 50% Final presentation and report
- 50% Final exam

To pass the course you need to collect at least a half of the coursework project points: **if you don’t do well on homeworks and the project -- you cannot pass the course!**

If you fulfil the above condition, the end mark will depend on your aggregate course completion percentage according to the following formula: $M = \text{ceil}(P/10)$ where $M$ is the final mark, $P$ the number of points you’ve got (0-100), and ceil() is an integer ceiling function.

**Policies**

**Plagiarism and cheating**

Cooperative work is an important part of learning; you are encouraged to study together, discuss the lectures, and discuss the software solutions. With the exception of your project, DO NOT:

- turn in duplicate work (no matter how small the shared part is)
- copy work (even one line) from another student's assignment or from a published source without citing the original material
- lend another student your assignment or look at someone else's assignment to fix your problem
- e-mail or transfer any of your homework solutions to another student or store your solutions on a computer to which another student in the class has access.

In addition, anyone caught cheating on the final exam will fail the course. The University of Ljubljana policy on academic honesty can be found here: [http://www.uni-lj.si/o_univerzi_v_ljubljani/organizacija__pravilniki_in_porocila/predpisi_statutUl_in_pravilniki/2013071214420651/](http://www.uni-lj.si/o_univerzi_v_ljubljani/organizacija__pravilniki_in_porocila/predpisi_statutUl_in_pravilniki/2013071214420651/) (in Slovenian, but Google translate does a good job in translating it to English). Note that cheating on the exam is considered a major breach of policy and can result in a suspension from the University.

Finally, if you are struggling with the course and need help, contact the instructor or the TA and we will do all that we can to help, including meeting outside of
regular office hours if need be, just DO NOT CHEAT.

**Project work**

In project work you are encouraged to collaborate, because this is what happens in the real world. However, we still need to evaluate your performance. It is your responsibility to make sure that each of the team members puts equal amount of effort into the project. You will be given a single grade for your project, and if your partner is slacking off - you will bear consequences, just like in a real company. Note, however, that we mentioned effort - we know that not all of you come with the same previous knowledge of computer science and that will be taken into account. Yet, everyone is expected to try hard. Through the project you will also develop skills that are necessary for team-work: managing and motivating your team mates, dividing tasks, etc. Please talk to your instructor or TA as soon as you spot problems in your team and we will help you sort them out.

**Attendance**

Attendance is required at all lectures. You need to ensure that you attend lab sessions where you will discuss how your project is going with the TA. You need to meet the TA at least ten times during the semester. The attendance is checked via sign up sheets. Therefore: you must sign in every time, and you may not sign in on behalf of another student, or ask someone else to sign in for you. In addition, you must take exams. There is only one exam per exam period, there are no make-up exams outside the scheduled periods.

**Online course materials**

The textbooks for the course are expensive, and to help you alleviate some of the expense we put a substantial amount of material on the course webpage. Please respect the authors and copyright holders, as well as the applicable laws, by not distributing the materials further, either online or offline. This holds for all the materials (presentation slides, articles, etc.) on the course webpage, but especially for chapters from course textbooks.

**Students with Disabilities**

If you are a student with a disability and would like to discuss special academic accommodations, please contact the instructor. In addition, the University of Ljubljana has adopted special guidelines regarding university procedures and the study process itself to ensure special needs students have equal rights and access to public information. Please contact Helena Zupan (phone: 01 476 81 80, e-mail: helena.zupan@fri.uni-lj.si ) who is in charge for handling such needs at FRI.