Survival Guide 02110

Inge Li Gørtz

Welcome to 02110. Here is a short survival guide for the course.

Structure

The course runs on Thursdays 8-12. Exercise class is from 8.00-10.00 and lectures from 10.00-12. There is a weekplan for each week containing information about literature for the week and exercises. The weekplans are placed under the week, where the lecture in the subject is given. Then you work with the exercises at the exercise class the week after.

Exercise class Thursday 8-10.30. The exercise classes will work as follows:

- *8.00-9.15 Group work.* Time to work on the exercises you couldn't solve at home. The TAs will be there to help you.
- *9.15-9.45 Walk-through of solutions to the exercises.* Together with the TA you will go through the solutions to the exercises in class. You are expected to have already solved the exercises and you should be prepared to present your solutions to the rest of the class.

Lectures Thursday from 10-12.

Textbook KT: "Algorithm Design" by Kleinberg ad Tardos. Additional literature will be supplied on CampusNet.

Mandatory assignments

In order to be allowed to participate in the written exam it is a requirement that you:

- get at least 40 point in the mandatory assignments from the week plans (more info below).
- implement (and pass the tests of) two of the implementation exercises on CodeJudge.

Each week from week 1 to 10 you will be asked to do written mandatory exercises and hand them in. These exercises will be corrected by the teaching assistants. It is a *requirement for participation in the exam* that you get at least 40 point in the exercises. Each weekly exercise can give up 20 points. 20 points is given for a perfect solution, and you can get anything between 0 and 20 points for a hand-in depending on the quality of your solution and your writing. The exercises do not count in the final grade for the course, but you have get at least 40 point in order to be allowed to participate in the exam. Start early, do not wait until the last 2 or 3 weeks before you hand in exercises. The homework on weekplan x is to be handed in online Sunday in week x+1 at 20:00 the latest. The deadline for handing in the home work must be respected.

Collaboration policy for mandatory exercises All mandatory exercises are subject to the following collaboration policy.

The exercises are individual. It is not allowed to collaborate on the exercises, except for discussing the text of the exercise with teachers and fellow students enrolled on the course in the same semester. Under no circumstances is it allowed to exchange, hand-over or in any other way communicate solutions or part of solutions to the exercises. It is not allowed to use solution from previous years, solutions from similar courses, or solutions found on the internet or elsewhere.

Exercises in CodeJudge

All exercises marked with [CJ] are implementation exercises and can be tested in CodeJudge. For each of these exercises, a detailed specification of the input/output and a java template can be found on CodeJudge. These exercises gives you an opportunity to implement some of the algorithms we work with. We recommend that you do these exercises after the other exercises on the weekplan.
It is a *requirement for participation in the exam*, that you pass at least two of the CodeJudge exercises. The deadline is Thursday in week 10.

Study tips

How do I prepare for class? You are expected to have looked at all exercises at home and to have solved most of them, and you should be prepared to present your solutions to the rest of the class. Two of the exercise classes will be in Danish.

We also assume that you read the material for the week's lecture. You can read it before or after the lecture, that's up to you. The lecture is meant to clarify and give an overview — it will not necessarily cover all material for the week in details.

It is a good idea to hand in as many homework assignments as possible. The questions will be similar to the exam questions, so it is a good opportunity to get some feedback on you writing.

How should I write my mandatory exercises? The ideal writing format for mandatory exercises is classical scientific writing, such as the writing found in the peer-reviewed articles listed as reading material for this course (not textbooks and other pedagogical material). One of the objectives of this course is to practice and learn this kind of writing. A few tips:

- Write things directly: Cut to the chase and avoid anything that is not essential. Test your own writing by answering the following question: Is this the shortest, clearest, and most direct exposition of my ideas/analysis/etc.?
- Add structure: Dont mix up description and analysis unless you know exactly what you are doing. For a data structure explain following things separately: The contents of the data structure, how to build it, how to query/update it, correctness, analysis of space, analysis of query/update time, and analysis of preprocessing time. For an algorithm explain separately what it does, correctness, analysis of time complexity, and analysis of space complexity.
- Be concise: Convoluted explanations, excessively long sentences, fancy wording, etc. have no in place scientific writing. Do not repeat the problem statement.
- Try to avoid pseudocode: Generally, aim for human readable description of algorithms that can easily and unambiguously be translated into code.
- Examples for support: Use figures and examples to illustrate key points of your algorithms and data structures.

Can I write my assignments in Danish? Ja. Du er meget velkommen til at aflevere på dansk.

Prerequisites

The course builds on 02105/02326 Algorithms and Data Structures I. You are expected to know the curriculum for 02105, which includes

- Basic algorithm analysis, asymptotic notation.
- Data structures: stacks, queues, linked lists, trees, heaps, priority queues, hash tables, union-find.
- Searching and sorting: binary search, heap sort, insertion sort, merge-sort.
- Graph algorithms: single source shortest paths (Dijkstra and SSSP in DAGs), Minimum spanning trees, topological sorting, Breadth first search, Depth first search, representation of graphs.