Welcome to Data Structures and Algorithms for Y!

The course (Y version) is intended for students other than the students of the University of Science. If you have completed the courses CS-A1110 Programming 1 and CS-A1120 Programming 2, you may want to take the course CS-A1140 Data Structures and Algorithms (see FAQ if you are uncertain about which course you should enroll). Tuition Y also suggests that the course's starting point is Basic Programming Y1 (Python).

The course has mandatory exercises as well as a voluntary project that can increase the course's rating. Exercises are done in a separate network environment and automatically receive feedback. Exercises can be done independently at your own pace within the deadlines. The project is done as a team work and returned electronically in three phases. The first two steps get feedback and the final step is evaluated.

Lectures and invoice support support the exercises and the project. It is worth doing some of the exercises in the course before making a bargain. Some of the exercises are done by programming the Python programming language so Python 3 should be installed in your work environment in accordance with tutorials. Help is provided by hourly assistants who are on call in the billing box.

Before starting the first lecture and starting the exercises, it is worthwhile to find a course brochure and to enroll in the course at WebOodi.

The course has plenty of material and tasks online. They are not here in MyCourses but in a learning environment called A +. The link to the material will be opened after the first lecture on 12.9.2017.

A + also has a separate discussion forum that is used on the course. MyCourses corresponds only to sending official bulletins (they should be subscribed to e-mail, so there's little to MC in MC).

F&Q

**FAQ in English**

Q: I read at some place that one can take this course also in English. What I need to do?

A: Non-Finnish speaking students may take this course by self-studying the material and assignments. Even though the course tutorial is in Finnish, you can find the assignments in English as well. In addition, the tutorial includes links to the textbook that is in English. Consult your peers or TAs in case you need help finding the assignments. During the registration, you can select the language between suomi/Finnish and ruotsi/Swedish (unfortunately there is no English option). Select suomi and send an email to

cs-a1141@aalto.fi. We maintain a mailing list for those taking the examination in English.

Note! This term we will be using Piazza for class discussion. The system is highly catered to getting you help fast and efficiently from classmates, and instructors. Rather than emailing questions to the teaching staff, I encourage you to post your questions on Piazza. Find our class page at: [https://piazza.com/aalto.fi/fall2017/csa1141/home](https://piazza.com/class/j7bj334vu3g6sm?cid=4)

Course Prerequisites

Q: I have completed CSE-A1110 Programming 1 (Scalalla), but I changed the degree program, which includes a CSE-A1141. Its prerequisite course has Python enabled. Can I attend the course this autumn even though I have not been to the Python Course? Or should I run the Python Course first?

A: There are two options in basic IT courses. Basically, either one should complete in full, ie CS-A1110 Programming 1, CS-A1120 Programming 2 and CS-A1140 Data Structures and Algorithms. If you have completed the A1120 then you can go straight to the course CS-A1140 for this course. If you run IT as a minor, it is recommended that you first read A1120 and then take the A1140 just then. Another option is to carry out the so-called. Y-courses, ie CS-A1111 Programming Basic Course Y1, CS-A1121 Basic Programming Y2 and CS-A1141 Data Structures and Algorithms Y. Basic Course Y1 is the other two precursor information. You may also try to attend a course at this course alongside this course, but it is not recommended.

Registration for the course

Q: I have the School of Science student and degree program is included in the CS-A1140. Can I, however, come up with the CS-A1141?

A: If you have already completed CSE-A1110 Programming 1, you will not be able to enter this course, but you will need to complete the CS-A1140 (but before CS-A1120). If you have not completed any Basic Programming Course, you can not come to either course. In this case, perform the CS-A1110 first. If you have completed a basic course of some other old programming (eg, CSE-A1111 but you have not completed the CSE-A1110), you may be able to take part in this course at a low cost. In that case, send an e-mail to cs-a1141@aalto.fi.

Replacement courses

Q: I am a switcher degree program and passed the old course T-106.1220 or 106.1223-T or T-106.1227. Should I run the CSE-A1141?

A: No need. Courses replace one another, so more can not be included in the degree.

Retaining parts

Q: How long will partial performances remain?

A: Contributions are valid for the course. They will not stay from one course to another. See. Aalto University's General Rules for Teaching and Studying:

Section 34 Validity of the studies

... the undergraduate studies are valid until the next course is held unless otherwise specified in the higher education degree or in the curriculum.

Q: I'm not getting enough points for weeks to get the course going. However, our group did a fairly good project work, so would it be possible to postpone the performance of a teamwork until next autumn?

A: The project is part of weekly exercises, that is, it is not an independent part, but part of the part. The subdivisions (and thus also the partial deliveries of part-performances) are valid for the course (see previous question). Part-performances are specific to the course, so they can no longer be supplemented except in exceptional cases after the course ends. They can not be moved to the next year either. In fact, it is unlikely that in the following year the course would have exactly the same share or partial performance, so giving a promise would be impossible.

Credits 5 cr

teaching Period

I - II

Workload by implementation

Lecture lessons 14 h (7 times), exercise sessions 22 h (11 times), group work 24 h, self study approx. 70 h (approx. 5.5 h / week, incl. Reading material reading, preparation for exams, etc.).

Learning outcomes

After completing the course you will be able to define, compare and implement basic data structures and algorithms, and name and select them eg for search structures, organization problems, and network scanning. In addition, you are able to identify and present a more detailed data structure or algorithm, and you can provide examples of their operation. You can also discuss other key data structures and algorithms using the industry's typical terminology.

Contents

Linear data structures, tree structures and nets. Search and Arrangement Methods. Basics of algorithmic analysis.

Implementation, methods and criteria

Exercises (individual homework assignments + project) and exam. The grade of the course is determined by half on the basis of the Exercises and the Half-Exam Score. Both must have an approved grade.

Exercises (H) and Exam (T) are evaluated on each scale from 0 to 5. The total value of the course is given by the formula

kas = min {1, H, T} \* round (0.5 \* H + 0.5 \* T).

Study Material

Finnish tutorial, English textbook and online exercises.

Substitutes

Replaces courses CSE-A1141, T-106.1223 and T-106.1227. NB! If you have completed a parallel course CS-A1140 or one of its replacement courses, you may not include this course anymore at your degree.

Prerequisites

CS-A1111 Programming Basic Course Y1

grading scale

0-5

Registration for exams

Language of Instruction FI. Finnish.

Course staff and contact information

Lectures: Ari Korhonen

Exercises: Timo Räsänen, Artturi Tilanterä and Saskia Kivistö. Assistants can be found on the invoice.

Email: cs-a1141 ät wave point fi

Reception times Match the time by email by sending mail to cs-a1141 ät aalto point fi.

More information

Detailed Prerequisites, Content, Learning Outcomes, and Time Use

Prerequisites

The course pre-requisite is CS-A1111 Programming Basic Course Y1. The following are some of the conceptual issues that should at least be managed well before entering the course. You can keep up to your memory by utilizing the lecture material and exercises of the pre-course. If the topic is not familiar, it is recommended that it be multiplied.

Programming, variables, expressions

truth values, lists, choice

code quality, refactoring, constants, visibility

translation

basic types, types of reference

loops, application area

strings

directories

the program's running time

tables

recursion

input and output streams, files.

Contents

The content of the course is divided into three parts, where "always essential material" describes the minimum requirements for access to knowledge and skills (Tasks A). At this level, the student must be able to explain the operation of said data structure or algorithm in words (as an indication of comprehension) and by using an example (code reading skills at a sufficient detail level). There should be so many details in the examples that the analytical characteristics of the data structure or algorithm are also apparent. In an example exercise, the starting point could be, for example, the program code of the express organizing method, from which the operation of that algorithm should be able to be explained using appropriate examples. The examples produced by the student should explain why. the worst case time of the algorithm is quadratic although it averages "n log n" in time.

However, the above is not enough for students who read more about computer science (the course is part of compulsory courses for a minor subject). If TRAK is a prerequisite for any future course, in addition to the aforementioned information and skills, it will also have to manage "often necessary material" (Tasks B). At this level, the student must also be able to produce code and thereby apply the learning they have learned to develop new simple algorithms. An example exercise could be, for example, writing a new algorithm for finding solid components in the network. This requires both a conceptual understanding (a coherent component of the network) and the application of a suitable algorithm (eg depth search) to solve a new problem. Also, the ability to compare two similar problem solving problems with each other is an important skill. An example could be, for example, a comparison of three different methods of organization (eg quick-ordering, inventorying and redrawing) on ​​the basis of criteria (eg stability, memory consumption and time complexity).

Especially for minor subject students, at least some things are recommended in the column "sometimes useful material" (Tasks C). The columns roughly correspond to the grades 1-2, 3 and 4-5, respectively.

Course Learning Outcomes

After completing this course ...

... always adopting the elements of the Essential Material (A) 1) define concepts such as minimal animation tree; 2) define abstract data types such as stack, queue, search structure, and priority queue; 3) describe their different ways of implementation; 4) to designate key data structures and algorithms for eg search structures, organizational problems and network scanning; 5) identify and present a more detailed data structure and algorithm; 6) provide examples of the operation of the given algorithm; and 7) discuss other key data structures and algorithms using scientific terms in the field (eg iso O)

... after adopting not only the above, but also the parts of the often necessary material (B) 1) comparing the implementations of data structures; 2) comparing algorithms with their analytical properties (such as performance rate); 3) to select from among the appropriate data structure and algorithm in different situations; 4) Implementation of data structures and algorithms for implementing inventory;

... after adopting some of the tools that are sometimes useful (C) 1) implement data structures and algorithms in several alternative ways; 2) combine data structures and algorithms to improve performance; 3) to compare the analytical properties of the various implementation methods both mathematically and empirically; 4) justify and choose the most suitable option for a specific application; 5) Modularly design and implement software by dividing larger software into smaller subdivisions that can be implemented with known data structures and algorithms.

Time use

In many cases, the subject of the course is cumulative (for example, downstream web algorithms rely on abstractions at the beginning of the course, such as a stack and queue, and to understand the operation of balanced search trees, a simpler binary search tree needs to be understood). Cumulativity is also reflected in the course structure (the courses are based on previous courses). Therefore, your target level should be set at a very early time to avoid problems later. In other words, afford sufficient time to complete the course throughout the autumn. Based on consistent time use, if 5 credits = 132 hours are divided into 13 weeks, then the course should be used for more than 10 hours per week. In fact, the number of hours per week may vary eg. because there is an exam week, where there are no deadlines and you may spend more time on other courses. It is recommended that you reserve the time for the calendar for several days a week.

|  |  |  |  |
| --- | --- | --- | --- |
| Arvosana | Taso A | Taso B | Taso C |
| 1 | 90 % | 10 % | 0 % |
| 2 | 100 % | 50 % | 0 % |
| 3 | 100 % | 100 % | 0 % |
| 4 | 100 % | 100 % | 25 % |
| 5 | 100 % | 100 % | 50 % |