NP-complete Problems at edX: Syllabus

July 27, 2018

Contents

1 Welcome!

Thank you for joining NP-complete Problems at edX!

Although many of the algorithms you've learned so far are applied in practice a lot, it turns out that the world is dominated by real-world problems without a known provably efficient algorithm. Many of these problems can be reduced to one of the classical problems called NP-complete problems which either cannot be solved by a polynomial algorithm or solving any one of them would win you a million dollars (see Millenium Prize Problems) and eternal worldwide fame for solving the main problem of computer science called P vs NP. It's good to know this before trying to solve a problem before the tomorrow's deadline :) Although these problems are very unlikely to be solvable efficiently in the nearest future, people always come up with various workarounds.

We will study the classical NP-complete problems and the reductions between them. You will practice solving large instances of some of these problems despite their hardness using very efficient specialized software based on tons of research in the area of NP-complete problems. We will show that some special cases on NP-complete problems can, in fact, be solved in polynomial time. We will then consider exact algorithms that find a solution much faster than the brute force algorithm. We will conclude with approximation algorithms that work in polynomial time and find a solution that is close to being optimal.

Programming challenges (available to verified learners only) are a very important part of this course. We believe that the main way to fully understand an algorithm is to implement it. If you're studying this course as a part of Algorithms and Data Structures MicroMasters, you're already familiar with the style of the programming challenges from the first course of the program Algorithmic Design and Techniques. However, if you're taking this class independently, we strongly recommend that before you start working on the programming challenges in this course, you first go through the videos and readings of the first week of the Algorithmic Design and Techniques class: they will guide you through the process of implementing a solution, testing it, finding bugs, fixing them and submitting the challenge. The second week of the also introduces big-O notation that is widely used in upper bounds on the running time of algorithms.

We look forward to seeing you in this class! We know it will make you a better programmer.

2 Who This Class Is For

Programmers with basic experience looking to understand the practical and conceptual underpinnings of algorithms, with the goal of becoming more effective software engineers. Computer science students and researchers as well as interdisciplinary students (studying electrical engineering, mathematics, bioinformatics, etc.) aiming to get more profound understanding of algorithms and hands-on experience implementing them and applying for real-world problems. Applicants who want to prepare for an interview in a

high-tech company.

3 Prerequisites

Basic knowledge of at least one programming language (`C`, `C++`, `Java`, `JavaScript`, `Python2`, `Python3`, `Scala`): loops, arrays, stacks, recursion. Basic knowledge of mathematics: proof by induction, proof by contradiction.

4 Learning Objectives

You will be able to apply the right algorithms and data structures in your day-to-day work and write programs that work in some cases many orders of magnitude faster. You'll be able to solve algorithmic problems like those used in the technical interviews at Google, Facebook, Microsoft, Yandex, etc. If you do data science, you'll be able to significantly increase the speed of some of your experiments.

5 Estimated Workload

We expect this course will take you 8–10 hours per week to complete.

6 Grading

Your grade will be composed out of the following components: two programming assignments (PA's) and the final proctored exam.

assignment	weight	# problems	# droppable
PA1: NP-complete Problems	30%	3	1
PA2: Coping with NP-completeness	30%	4	2
Final exam	40%	2	0

The passing thresholds are $\geq 70\%$ for C, $\geq 80\%$ for B, and $\geq 90\%$ for A. To receive a certificate, you need to get at least C. Passing Grade: you must score 70% or above to pass the course. To get a university course credit, you must score at least 80%.

7 Deadlines

This course is self-paced, so there are no deadlines for any specific assignment. This course will be open until August 01, 2019. Shortly after that we expect to re-open the course, self-paced, with updates.

8 Verified Certificate

There are a number of differences for verified learners compared with those just wishing to audit the course.

- **Programming challenges.** Verified learners have access to programming challenges, the most important part of the course. To solve a programming challenge, one needs to implement a reliable (that works correctly on all test cases) and efficient (that works in less than one second even on massive datasets) program. Writing fast and correct programs is an important skill of a software engineer.
- **Official proof of completion.** By completing the course, verified learners receive an easily shareable official certificate. To earn a verified certificate, you need to be enrolled as part of the verified track, complete identity verification before you take the proctored final exam, and earn a passing grade. If you are auditing the course, you will not receive a certificate.
- **University course credit.** This class is a part of MicroMasters program "Algorithms and Data Structures". By completing the MicroMasters program, you indicate to employers and hiring personnel that you have made a significant investment in completing rigorous, Masters-level courses and content.

  Learners who successfully earn the Algorithms and Data Structures MicroMasters Credential are eligible to apply for admission to the School of Individualized Study (SOIS) Master of Science in Professional Studies at Rochester Institute of Technology.

  If a learner applies for admission to the SOIC Master of Science in Professional Studies program at Rochester Institute of Technology, and is accepted, the MicroMasters Credential will count towards 25% of the coursework required by this program.

  To receive a MicroMasters certificate, you must receive verified certificates in all eight courses in the program. This course is worth 20 credits. The other courses in the MicroMasters program:

  - Algorithmic Design and Techniques (15 credits)
  - Data Structures Fundamentals (15 credits)
  - NP-Complete Problems (10 credits)
  - String Processing and Pattern Matching Algorithms (10 credits)
  - Dynamic Programming and Its Applications In Genomics and Machine Learning (10 credits)
  - Applications of Graph Algorithms in Genome Sequencing (10 credits)
  - Capstone Project in Genome Assembly (10 credits)
- **Proven motivator to complete the course.** MOOCs data tells that verified learners complete courses at a much higher rate than audit learners.

9 Forum

The forum can be found under the Discussion tab from the course home page.

We encourage you to visit the forum each time you work on a programming challenge. We've set up a separate thread for every programming challenge. At this thread, you may ask questions as well as help other learners and learn from other learners.

The instructors and TA's are going to constantly monitor the forums to help the learners to overcome their difficulties as quickly as possible.

Forum Rules

- Follow etiquette rules: respect other learners, make your post valuable for others (in particular, before asking check whether someone else has already asked it; keep your post as short as possible). See more common sense rules to follow.
- Please do not post solutions of programming challenges or their algorithmic parts (this violates the edX honor code), even if they do not pass the tests.
- We encourage you to post starter files for various programming languages. A starter file should only read the input data and write the output data.
- If your first attempt to solve a programming challenge failed, and then you debugged and fixed your program, you may want to share this bug. Examples: "Remember to use `//` for integer division for `Python3`", "Remember that a class name and a file name should match for `Java`", "Remember to use the `long long` type if you are dealing with large numbers for `C++`", but please do not post the code, just mention the "idea" of the bug.
- You may want also to help other learners by posting small tests that can help to catch a bug. In this case, please comment how did you come up with the test, so that others can learn from your creativity.
- Please note that instructors are not going to reply to the following typical questions.
  - *"My program runs fine on my local machine, but fails in the grader. Could you fix the grader?"*
    All the learners' computers are slightly different, so the only way to make grading fully objective is to grade all solutions on the same server. Sometimes, the result of compiling and running the same program is different on different computers because of different compilers, different compiler settings, different operating system or just some bugs in the program code that lead to undefined behavior. However, it is always possible to write a correct program that works correctly both on your local machine and in our grader. You will have to write such a program to pass. We do our best to specify all the important parameters of the grader that we know of here.

– *"I use exactly the same compiler and flags as in the grader, but my program has different outputs on my machine and in the grader. Could you check the grader?"*
This usually happens with buggy programs that run into undefined behavior.

– *"Any hints about test 42?"*
Recall that we hide the test cases intentionally. Please test and stress test you program to fix it. If you have already tested a lot (considered all corner cases that you can imagine, constructed a set of manual test cases, applied stress testing), but your program still fails and you are stuck, try to ask for help on the forum. We encourage you to do this by first explaining what kind of corner cases you have already considered (it may happen that when writing such a post you will realize that you missed some corner cases!) and only then asking other learners to give you more ideas for tests cases.

– *"How to compile/run a starter file?"*
Please note that though this class has many programming exercises, it is not a programming class. We expect you to know how to program in a programming language of your choice.