

Course introduction: Computational complexity, undecidable problems, and optimization methods

Petr Kurapov

petr.kurapov@gmail.com

2024

About this course

- Computational complexity theory (~4 lectures):
 - Complexity classes, theorems, Turing machines and existential questions (Based on: Computational Complexity: A Modern Approach)
- Algorithm's & data structure access complexity by example (~6 lectures):
 - Sorting, heaps, trees, sparse graphs (Based on: Introduction to Algorithms + common sense)
- NP-complete/hard problem solving (~2 lectures):
 - Dynamic programming, B&B, annealing etc. (Based on: MIT 6.172)

Why?

- Write high-level language code ~~and be happy about performance~~
- Recognize *really* hard problems & be able to apply basic solution approaches
- Answer weird job interview questions

Your input is welcome – you may request a topic that aligns with the course

About this course

Tailored by:

- Optimizing compilers
 - Classical solutions and modern ML ecosystem
- Database management systems
 - Efficient data structures and synchronization primitives
- ML and data preparation
 - New types of workloads, heterogeneous parallelism
- Explosion of hardware
 - Many types of compute, memory, storage, and networking
- Academia research

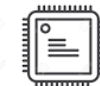


HyPer



MODIN

PyTorch



Credit

Part 1 (60 out of 100) – homework

- At the end of each of the three modules
- Each gives you 20 points

Part 2 (30 out of 100) – exam

Part 3 (10 out of 100) – reading assignments


Your resulting grade is determined by the number of points divided by 10.

Your credit schema will be defined upon problem submission

Warning: late submissions will suffer penalties

Warning: plagiarism will not be tolerated. If unsure – ask Peter.

Reading assignments

- Some of the course materials are heavily based on recent (or not so much) research
- Additional reading will be marked by a special sign: 
- There will be several must-read papers that will contribute to your final credit
- Reading assignment assumes writing a short summary of the paper with your own words and thoughts

Warning: plagiarism will not be tolerated.

Course deadlines

1. 1st homework & reading assignment **11:59 pm Oct 4th**
2. 2nd homework & reading assignment **11:59 pm Nov 15th**
3. 3rd homework & reading assignment **11:59 pm Dec 6th**

Administrativa

Office hours:

- 4-5pm every Friday + any spare time within schedule
- Schedule it early!

A Google doc with grading will be shared shortly

The communication channel is discord.

Questions?

Quiz time

```
vector<string> strings = ReadStrings();  
vector<string> strings_to_find = ReadStrings();  
auto result = search(begin(strings), end(strings),  
                     begin(strings_to_find),  
                     end(strings_to_find));
```

Quiz time

```
vector<string> strings = ReadStrings();  
set<string> chosen;  
remove_copy_if(begin(strings), end(strings),  
               inserter(chosen, chosen.begin()),  
               [](const string& s){return s[0]=='C';});
```

Copies elements from the range [first, last), to another range beginning at d_first, omitting the elements which satisfy specific criteria.

BACKUP

Credit options

Team project

- NP optimization problem solution
- Collaborative work: review, PRs, standard opensource flow
- (2020): Knight tour problem

Why do?

- Familiarize with standard tools
- Practice smart brute-force techniques/approximations

Solo project

- Low level optimization problem
- (2019): Tuning a small piece of binary code

Why do?

- Familiarize with low level optimizations (i.e.. vectorization)
- Practice benchmarking & performance results stabilization

Credit options

Team project

- NP optimization problem solution
- Collaborative work: review, PRs,

Solo project

- Low level optimization problem
- (2019): Tuning a small piece of

Example projects: database physical plan optimization, primitives optimization for heterogeneous execution, factor huge composite numbers

- Familiarize with standard tools
- Practice smart brute-force techniques

- Familiarize with low level optimizations (i.e.. vectorization)
- Practice benchmarking & performance results stabilization