Aalto University
School of Science

CS-E4530 Computational Complexity Theory

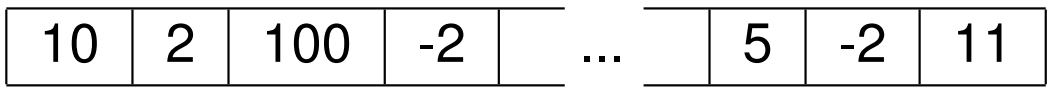
Lecturer: Prof. Pekka Orponen

Aalto University
School of Science
Department of Computer Science

Spring 2019

For a computer, how hard is it ...

to sort an array of integers?



For a computer, how hard is it ...

to verify that a C++ program can never segfault?

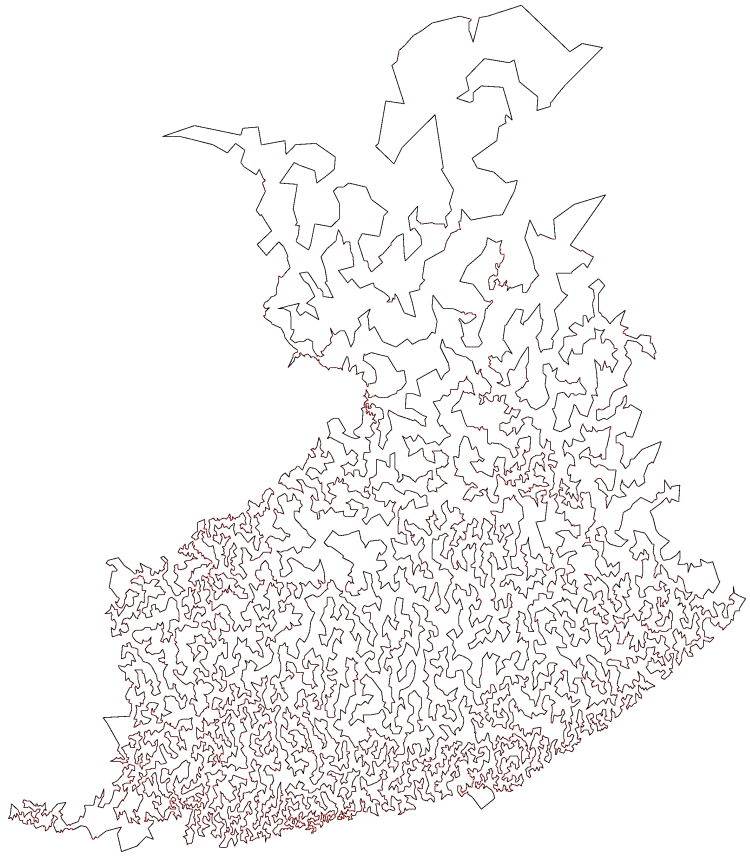
```
...
    for(unsigned int i = cell->length; i > 0; i--, ep++) {
      if((int)(*ep) > info.split_element &&
         *ep < next_split_element &&
         best_path_orbits.is_minimal_representative(*ep) &&
         (!opt_use_long_prune ||
          info.long_prune_redundant.find(*ep) ==
          info.long_prune_redundant.end())) {
        next_split_element = *ep;
        next_split_element_pos = ep;
      }
    }
...
```

For a computer, how hard is it ...

to solve sudokus?

10		16			12				15				4	5	1
	11		14	13			5	10				16			
			12									11			
	1		9			7	4			11	8	13		12	
5	10	14				11				9		3		4	
		9	7		4	6					15	1	11	13	16
				16		5	3						2	15	9
			6				7	2							
14		13		1		2		9		16		8	6		
16				7	14	9		8	1		2	5			
2		8			6	4		13	3		5	14		1	
			4								7				
			16	14						1		12			7
			11						14	5					
		2			10		6	11	7		13	9	5		
3		12	15									2		10	

to find shortest Travelling Salesperson tours?



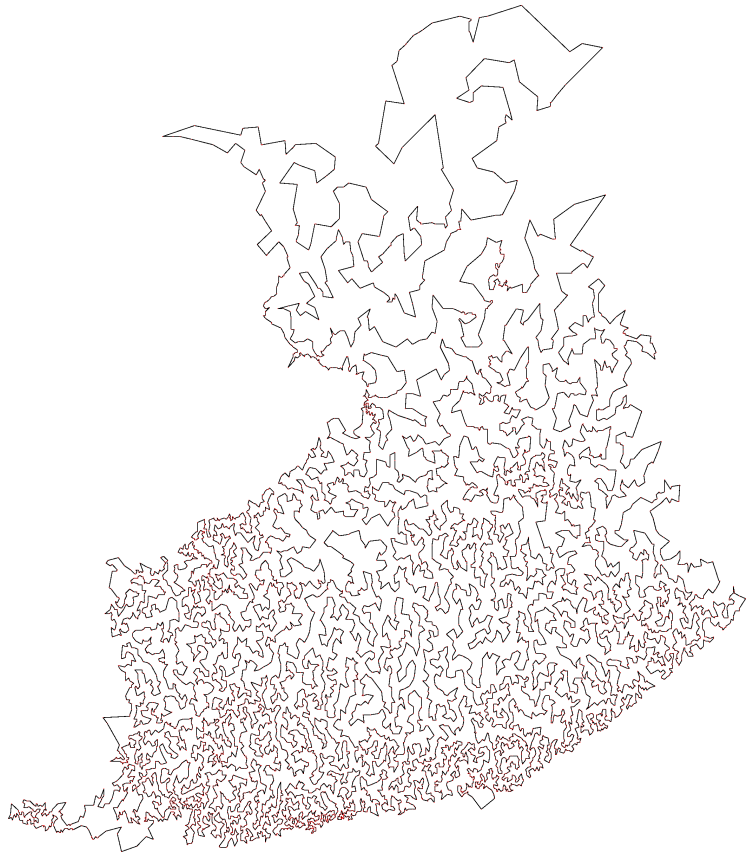
http://www.math.uwaterloo.ca/tsp/world/countries.html

For a computer, how hard is it ...

to **efficiently** solve sudokus?

10		16			12				15				4	5	1
	11		14	13			5	10				16			
			12									11			
	1		9			7	4			11	8	13		12	
5	10	14				11				9		3		4	
		9	7		4	6					15	1	11	13	16
				16		5	3						2	15	9
			6				7	2							
14		13		1		2		9		16		8	6		
16				7	14	9		8	1		2	5			
2		8			6	4		13	3		5	14		1	
			4								7				
			16	14						1		12			7
			11						14	5					
		2			10		6	11	7		13	9	5		
3		12	15									2		10	

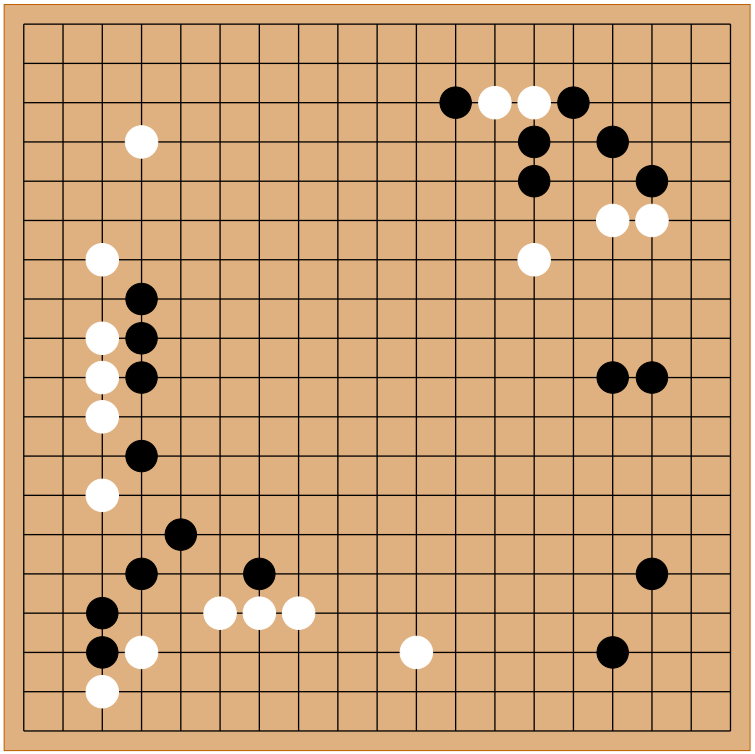
to **efficiently** find shortest Travelling Salesperson tours?



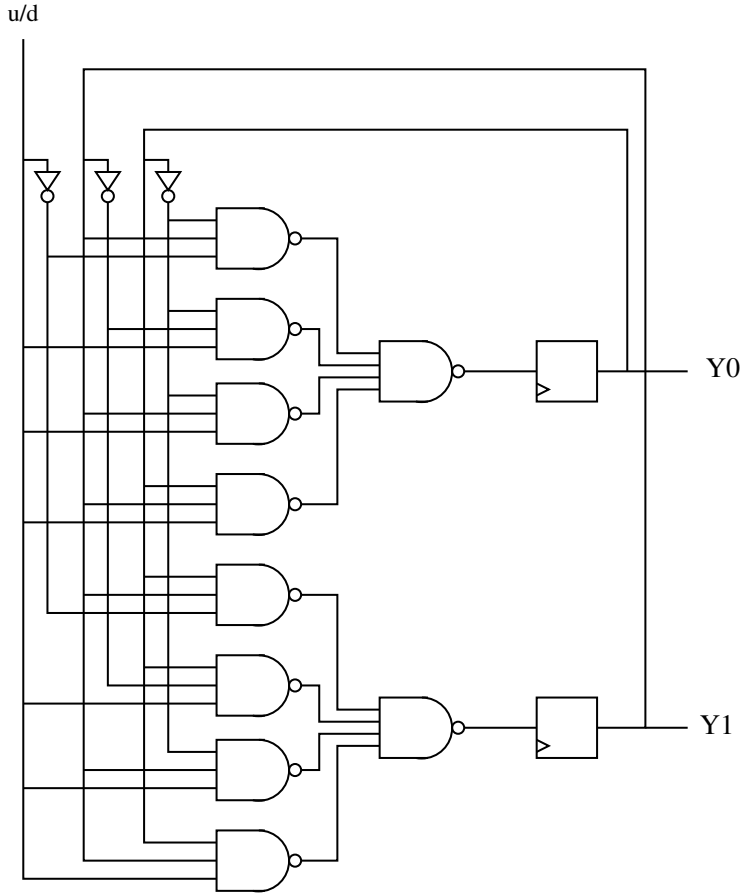
http://www.math.uwaterloo.ca/tsp/world/countries.html

For a computer, how hard is it ...

to determine whether
the white player
has a winning strategy
in a given Go configuration?



to deduce whether an output
line of a sequential circuit can
ever be active?



Weekly Sessions and Course Personnel

- **Lectures:** Mondays & Wednesdays 10–12, T3
- **Lecturer:** Prof. Pekka Orponen
- **Tutorials:** Wednesdays 12–14, B337
  First tutorial Wed 9 January
- **Tutorial assistants:**
  Dr. Juho Hirvonen and Dr. Gorav Jindal
- **TA office hours:** Fridays 13–14 at the respective offices (B314, C214).
- **Course home page:**
  - `https://mycourses.aalto.fi/course/view.php?id=20593`
  - Lecture slides, homework assignments, current info etc. uploaded as the course progresses.

General Themes

- Concepts and phenomena of computational complexity
- Identification of computationally hard problems
- Classification of problems according to their complexity
- Structure of the universe of complexity classes

Topics

- Models of computation: Turing machines, RAM machines, Boolean circuits
- Modes of computation: deterministic, nondeterministic, parallel, randomised, alternating
- Basic notions of computational complexity: complexity measures and complexity classes, hierarchy theorems, reductions between problems, completeness
- Central complexity classes: P, NP, PSPACE, NC, polynomial time hierarchy, etc.
- Design of completeness proofs, especially for NP-completeness
- Applications and extensions: approximability, counting, games, cryptographic protocols

Material: S. Arora & N. Barak, *Computational Complexity*, Cambridge UP, 2009. Draft online at `http://theory.cs.princeton.edu/complexity/`.
Prerequisites: Course CS-C2150 Theoretical Computer Science or similar.

Learning Outcomes

- Once you have taken the course, you master the key complexity classes, their underlying models of computation, and relationships.
- You are able to formalise and abstract from a given computational task relevant computational problems and argue that they belong to appropriate complexity classes.
- You understand the concept of reductions and how it can be used to order problems by their computational complexity. You are able to show using reductions that a problem is complete for a central complexity class (such as NP) and you understand the importance and implications of such a result.
- You are familiar with the concepts of randomised, approximation, and parallel computation and aware of related complexity classes and their relation to other complexity classes and their models of computation.

Exams, Homework and Grading

- The course has two midterm exams, graded at 10 points and 30 points.
- In addition to the midterms, there are 10 weekly homework problem sets. Each problem set counts for 6 points, for a maximum of 60 points.
- The points accumulated from the exams and the homework problems determine the final grade as follows:
  1/5: 40 points
  2/5: 50 points
  3/5: 60 points
  4/5: 70 points
  5/5: 80 points.
- Note also that you *must register for the course* in Oodi.

Tutorials and Homework

- The weekly problem sheets are published on MyCourses each Monday.
- Each sheet contains 5-6 problems pertaining to material covered at the lectures in that week.
- The first 2-3 exercises are simple warmup problems that will be discussed and solved at the Wed tutorial *in the same week*.
- The last 3 are homework graded at 0–2 points per problem.
- You should return your solutions to these problems via MyCourses by *midnight Tuesday the following week*.
- **Code of Conduct:** You are allowed and encouraged to work together with your fellow students on the homework problems, and you can get help from the teaching assistants during the weekly tutorial sessions and their office hours. However, *you must write up your final solutions to the homework problems by yourself*, without written notes from such background discussions.