Computational Complexity

Lecture 1

February 5, 2020

Universiteit van Amsterdam

Practical Information

- Ronald de Haan (`R.deHaan@uva.nl`)
- Boas Kluiving (`R.B.Kluiving@uva.nl`)
- Course web page: `https://staff.science.uva.nl/r.dehaan/complexity2020/`
- Canvas page: `https://canvas.uva.nl/courses/10928`
- Book: *Computational Complexity: A Modern Approach*, by Sanjeev Arora & Boaz Barak, 2009. (`http://www.cs.princeton.edu/theory/complexity/`)

What is Computational Complexity?

- The study of what you can compute with limited resources
  - Resources, e.g.: time, memory space, random bits
- Computability theory tells us what can be computed in principle

  Computational complexity theory tells us what can be computed realistically
- Central notions:
  - The resource use of an algorithm, in the worst case
  - Computational problems, and algorithms solving these problems
  - How does the resource use of algorithms scale?
  - ...

What is Computational Complexity?

- Distinguish different degrees of computational difficulty:

  different complexity classes

  (a whole zoo of complexity classes:
  `https://complexityzoo.uwaterloo.ca/Complexity_Zoo`)

What is Computational Complexity?

- Distinguish different degrees of computational difficulty:

  different complexity classes

  (a whole zoo of complexity classes:
  `https://complexityzoo.uwaterloo.ca/Complexity_Zoo`)

- Central question: the P versus NP problem
  (one of the $1 Million *Millennium Prize Problems*)

Relation to Other Fields

- Computation plays a role nearly everywhere..
- Therefore, computational complexity is relevant for many areas; for example:
  - Computer science
  - Cryptography
  - Economics, game theory
  - Artificial intelligence
  - Biology
  - Scheduling, vehicle routing
  - ...

Some courses that are related

- Recursion Theory
- Kolmogorov Complexity
- Knowledge Representation and Reasoning
- Quantum Computing
- Machine Learning Theory
- Computational Social Choice

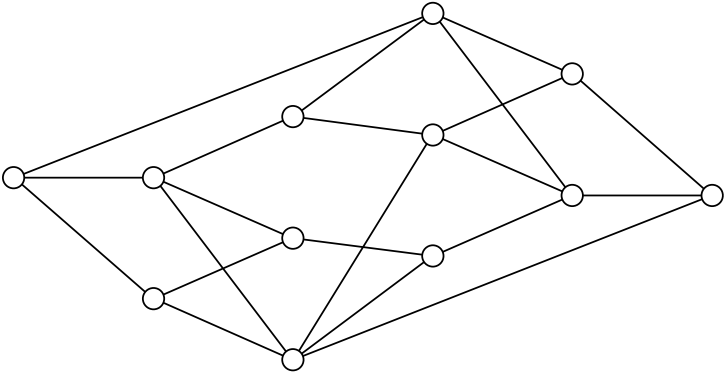
Illustration: Graph k-Coloring

- You are given an undirect graph
  - Nodes / vertices, with edges between them.
- The task is to color each node with a color in $\{1, 2, \ldots, k\}$ so that no two connected nodes have the same color

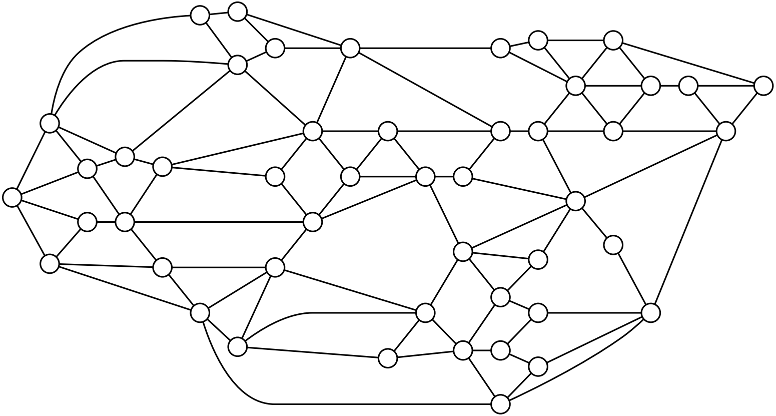
Illustration: Graph k-Coloring

- You are given an undirect graph
  - Nodes / vertices, with edges between them.
- The task is to color each node with a color in $\{1, 2, \ldots, k\}$ so that no two connected nodes have the same color
- This can be used to model many applications
  - For example, nodes are regions with their own radio station, colors are radio frequencies, and two nodes are connected if the regions border each other
  - The task is to assign radio frequencies without conflict (in the border areas)

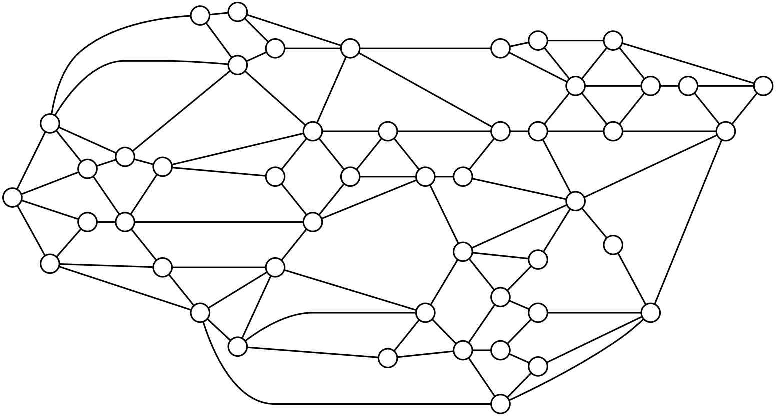
Color this graph with 2 colors!



Color this graph with 2 colors!



Now, color this graph with 3 colors!



Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2		
5		
10		
20		
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	
5		
10		
20		
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5		
10		
20		
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	
10		
20		
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10		
20		
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	
20		
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20		
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
50		
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
50	0.25 sec	
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
50	0.25 sec	8.4 centuries
100		
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
50	0.25 sec	8.4 centuries
100	1.00 sec	
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n

- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
50	0.25 sec	8.4 centuries
100	1.00 sec	9.4×10^{17} years
1000		

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
50	0.25 sec	8.4 centuries
100	1.00 sec	9.4×10^{17} years
1000	1.67 min	

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10.000 steps per second):

n	n^2 steps	2^n steps
2	0.02 msec	0.02 msec
5	0.15 msec	0.19 msec
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
50	0.25 sec	8.4 centuries
100	1.00 sec	9.4×10^{17} years
1000	1.67 min	7.9×10^{288} years

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10^{10} steps per second):

n	n^2 steps	2^n steps
2	0.00000002 msec	0.00000002 msec
5	0.00000015 msec	0.00000019 msec
10	0.00001 msec	0.0001 msec
20	0.00004 msec	0.10 msec
50	0.00025 msec	31.3 hours
100	0.001 msec	9.4×10^{11} years
1000	0.100 msec	7.9×10^{282} years

Quadratic vs. Exponential

- Important difference between algorithms that run in time, say, n^2 vs. algorithms that run in time, say, 2^n
- Illustration (time needed for 10^{10} steps per second):

n	n^2 steps	2^n steps
2	0.00000002 msec	0.00000002 msec
5	0.00000015 msec	0.00000019 msec
10	0.00001 msec	0.0001 msec
20	0.00004 msec	0.10 msec
50	0.00025 msec	31.3 hours
100	0.001 msec	9.4×10^{11} years
1000	0.100 msec	7.9×10^{282} years

- # of atoms in universe $\approx 10^{80}$