Complexity Theory

Jan Křetínský

Chair for Foundations of Software Reliability and Theoretical Computer Science
Technical University of Munich

Summer 2019

Partially based on slides by Jörg Kreiker

Lecture 1

Introduction

Agenda

- computational complexity and two problems
- your background and expectations
- organization
- basic concepts
- teaser
- summary

Computational Complexity

- quantifying the efficiency of computations
- not: computability, descriptive complexity, . . .
- computation: computing a function $f : \{0,1\}^* \to \{0,1\}^*$
  - everything else matter of encoding
  - model of computation?
- efficiency: how many resources used by computation
  - time: number of basic operations with respect to input size
  - space: memory usage

Dinner Party

Example (Dinner Party)

You want to throw a dinner party. You have a list of pairs of friends who do not get along. What is the largest party you can throw such that you do not invite any two who don't get along?

Dinner Party

Example (Dinner Party)

You want to throw a dinner party. You have a list of pairs of friends who do not get along. What is the largest party you can throw such that you do not invite any two who don't get along?

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

Dinner Party

Example (Dinner Party)

You want to throw a dinner party. You have a list of pairs of friends who do not get along. What is the largest party you can throw such that you do not invite any two who don't get along?

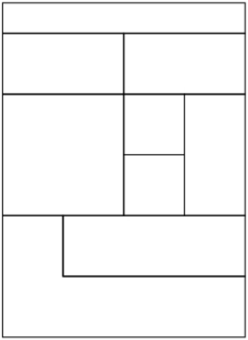
person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

- largest party?
- naive computation
  - check all sets of people for compatibility
  - number of subsets of n element set is 2^n
  - intractable
- can we do better?
- observation: for a given set compatibility checking is easy

Map Coloring

Example (Map Coloring)

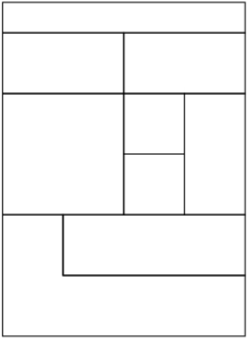
Can you color a map with three different colors, such that no pair of adjacent countries has the same color. Countries are adjacent if they have a non-zero length, shared border.



Map Coloring

Example (Map Coloring)

Can you color a map with three different colors, such that no pair of adjacent countries has the same color. Countries are adjacent if they have a non-zero length, shared border.



- naive algorithm: try all colorings and check
- number of 3-colorings for n countries: 3^n
- can we do better?
- observation: for a given coloring compatibilty checking is easy

What about you?

- What do you expect?
- What do you already know about complexity?
- Immediate feedback

Organization

- lecture in English
- course website:
  http://www7.in.tum.de/um/courses/complexity/SS19/
- concentrated into the first part of the semester, in 03.09.014
  - (reserved slot Monday 14-16)
  - Tuesday 10:05-11:35 and 12:25-13:55
  - Wednesday 8:25-9:55
  - Friday 12:05-13:35 and 14:00-15:30
- tutor: Mikhail Raskin
- weekly exercise sheets, not mandatory
- written or oral exam, depending on number of students
- bonus
  - several mini-tests during lectures (un-announced, cover 2-4 lectures)
  - self-assessment and feedback to us
  - if C is ratio of correct answers, exam bonus computed by

$$\frac{\lceil 5C - 1 \rceil}{2}$$

Literature

- lecture based on Computational Complexity: A Modern Approach by **Sanjeev Arora** and **Boaz Barak**
- book website:
  `http://www.cs.princeton.edu/theory/complexity/`
- useful links plus freely available draft
- lecture is self-contained
- more recommended reading on course website, e.g. Introduction to the Theory of Computation by **Michael Sipser**

Agenda

- computational complexity and two problems ✓
- your background and expectations ✓
- organization ✓
- basic concepts
- teaser
- summary

Prerequisites

- sets, relations, functions
- formal languages
- Turing machines
- graphs and algorithms on graphs
- little probability theory
- Landau symbols

Landau symbols

- characterize asymptotic behavior of functions (on integers, reals)
- ignore constant factors
- useful to talk about resource usage

Landau symbols

- characterize asymptotic behavior of functions (on integers, reals)
- ignore constant factors
- useful to talk about resource usage
- upper bound: $f \in \mathcal{O}(g)$ defined by
  $\exists c > 0.\ \exists n_0 > 0.\ \forall n > n_0.\ f(n) \leq c \cdot g(n)$
- dominated by: $f \in o(g)$ defined by $\forall \varepsilon > 0.\ \exists n_0 > 0.\ \forall n > n_0.\ \frac{f(n)}{g(n)} < \varepsilon$
- lower bound: $f \in \Omega(g)$ iff $g \in \mathcal{O}(f)$
- tight bound: $f \in \Theta(g)$ iff $f \in \mathcal{O}(g)$ and $f \in \Omega(g)$
- dominating: $f \in \omega(g)$ iff $g \in o(f)$

Intractability

Polynomial

versus

Exponential

- computations using exponential time or space intractable for all but the smallest inputs
- for a map with 200 countries: app. $2.66 \cdot 10^{95}$ 3-colorings
- atoms in the universe (wikipedia): $8 \cdot 10^{80}$
- computational complexity: tractable vs. intractable
- tractable: problems with runtimes $\bigcup_{p>0} \mathcal{O}(n^p)$
- intractable: problems with worse runtimes, e.g. $2^{\Omega(n)}$
- independent of hardware

What about our examples?

- dinner party problem tractable?
- map coloring problem tractable?
- lower bounds on time/space consumption
- upper bounds on time/space consumption
- which is harder?

Dinner Party

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

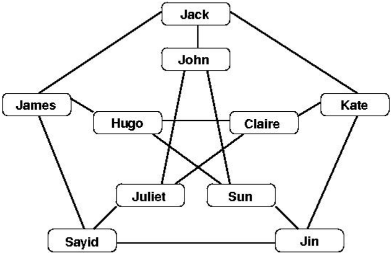
Dinner Party

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent

Dinner Party

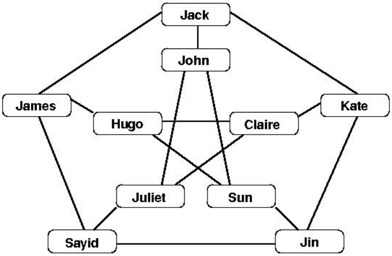
person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate



- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent

Dinner Party

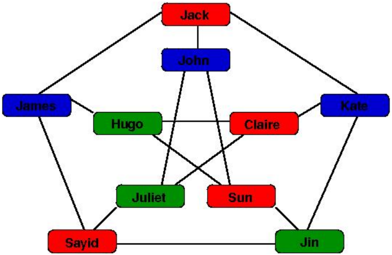
person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate



- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent
- the independent set problem: Indset
- probably not tractable, no algorithm better than naive one known

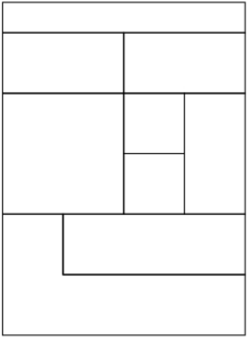
Dinner Party

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

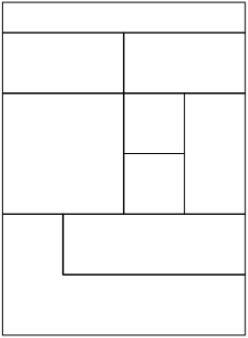


- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent
- the independent set problem: Indset
- probably not tractable, no algorithm better than naive one known
- here: maximal independent set of size 4

Map Coloring

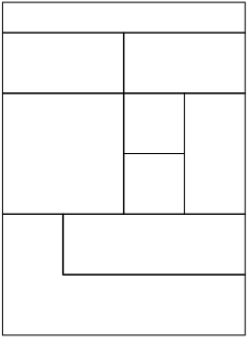


Map Coloring



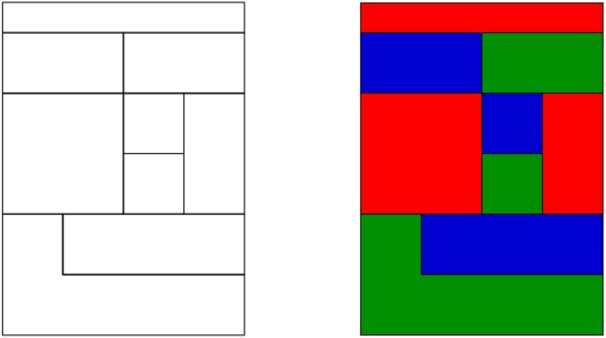
- really a graph problem
- each country a node, each border an edge
- color each node such that no two adjacent nodes have same color

Map Coloring



- really a graph problem
- each country a node, each border an edge
- color each node such that no two adjacent nodes have same color
- the three coloring problem: 3−Coloring
- probably not tractable, no algorithm better than naive one known

Map Coloring



- really a graph problem
- each country a node, each border an edge
- color each node such that no two adjacent nodes have same color
- the three coloring problem: $3-$Coloring
- probably not tractable, no algorithm better than naive one known
- here: answer is yes

Bounds

- upper bounds
  - time (naive algorithm): $2^{\mathcal{O}(n)}$ for n persons/countries
  - space (naive algorith): $\mathcal{O}(n^p)$ for n persons/countries and a small p

Bounds

- upper bounds
  - time (naive algorithm): $2^{\mathcal{O}(n)}$ for n persons/countries
  - space (naive algorith): $\mathcal{O}(n^p)$ for n persons/countries and a small p
- lower bounds
  - very little known
  - difficult because of infinitely many algorithms
  - both problems could have a linear time and a logarithmic space algorithm
  - but not simultaneously

Which is harder?

- instead of tight bounds say which problem is harder
- $\Rightarrow$ reductions

Which is harder?

- instead of tight bounds say which problem is harder
- ⇒ reductions

IF there is an efficient procedure for *B* using a procedure for *A*

THEN *B* cannot be radically harder than *A*

notation: $B \leq A$

Which is harder?

- instead of tight bounds say which problem is harder
- $\Rightarrow$ reductions

IF there is an efficient procedure for B using a procedure for A

THEN B cannot be radically harder than A

notation: $B \leq A$

Applications:

- **IF**
  - there is an efficient procedure for problem A and
  - $B \leq A$

  THEN there is an efficient procedure for problem B
- **IF**
  - there is no efficient procedure for problem B and
  - $B \leq A$

  THEN there is no efficient procedure for problem A

3−Coloring $\leq$ Indset

How can we solve 3−Coloring using an algorithm to solve Indset?

$3-$Coloring $\leq$ Indset

How can we solve $3-$Coloring using an algorithm to solve Indset?

- triplicate the original graph (V, E) into $(V \times \{1, 2, 3\}, E')$ where

$$
\begin{aligned}
E' = & \{((v, i), (w, i)) \mid (v, w) \in E, i \in \{1, 2, 3\}\} \cup \\
& \{((v, i), (v, j)) \mid v \in V, i \neq j \in \{1, 2, 3\}\}
\end{aligned}
$$

efficient

$3-\text{Coloring} \leq \text{Indset}$

How can we solve $3-$Coloring using an algorithm to solve Indset?

- triplicate the original graph (V, E) into $(V \times \{1, 2, 3\}, E')$ where

$$\begin{aligned} E' = & \{((v, i), (w, i)) \mid (v, w) \in E, i \in \{1, 2, 3\}\} \cup \\ & \{((v, i), (v, j)) \mid v \in V, i \neq j \in \{1, 2, 3\}\} \end{aligned}$$

efficient

- check whether there is an independent set of size $|V|$

assumed efficient

$3-$Coloring $\leq$ Indset

How can we solve $3-$Coloring using an algorithm to solve Indset?

- triplicate the original graph (V, E) into $(V \times \{1,2,3\}, E')$ where

$$E' = \{((v,i),(w,i)) \mid (v,w) \in E, i \in \{1,2,3\}\} \cup \\ \{((v,i),(v,j)) \mid v \in V, i \neq j \in \{1,2,3\}\}$$

efficient

- check whether there is an independent set of size $|V|$

assumed efficient

Need to ensure: procedure returns **yes** if and only if the graph is 3-colorable.

Polynomial certificates: NP

- whole class of problems can be reduced to Indset
- all of them have polynomially checkable certificates
- characterizes (in)famous class **NP**
- all problems in **NP** reducible to Indset makes Indset **NP**-hard.
- 3−Coloring also **NP**-hard
- no polynomial-time algorithms known for **NP**-hard problems
- not all problems have polynomial certificates, e.g. winning strategy in chess

Agenda

- computational complexity and two problems ✓
- your background and expectations ✓
- organization ✓
- basic concepts ✓
- teaser
- summary

Lots of things to explore

- precise definition of computational model and resources
- problems with polynomial time checkable certificates
- space classes
- approximations
- hierarchies (polynomial, time/space tradeoffs)
- randomization
- parallelism
- average case complexities
- probabilistically checkable proofs
- (quantum computing)
- (logical characterizations of complexity classes)
- a bag of proof techniques

What have we learnt?

- polynomial $\sim$ tractable; exponential $\sim$ intractable
- lower bounds hard to come by
- reductions key to establish relations among (classes of problems)
- **NP**: polynomially checkable certificates
- Indset $\in$ **NP**, 3−Coloring $\in$ **NP**