Theory of Computer Science

E5. Some NP-Complete Problems, Part II

Malte Helmert

University of Basel

May 31, 2017

Overview: Course

contents of this course:

- logic ✓
  - ▷ How can knowledge be represented?
    How can reasoning be automated?
- automata theory and formal languages ✓
  - ▷ What is a computation?
- computability theory ✓
  - ▷ What can be computed at all?
- complexity theory
  - ▷ What can be computed efficiently?

Overview: Complexity Theory

Complexity Theory

Further Reading (German)

Literature for this Chapter (German)

Theoretische Informatik – kurz gefasst
by Uwe Schöning (5th edition)

- Chapter 3.3

Further Reading (English)

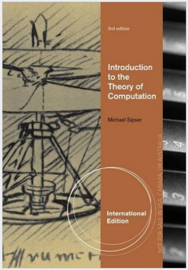
Literature for this Chapter (English)

Introduction to the Theory of Computation
by Michael Sipser (3rd edition)

- Chapter 7.5

Note:

- Sipser does not cover all problems that we do.

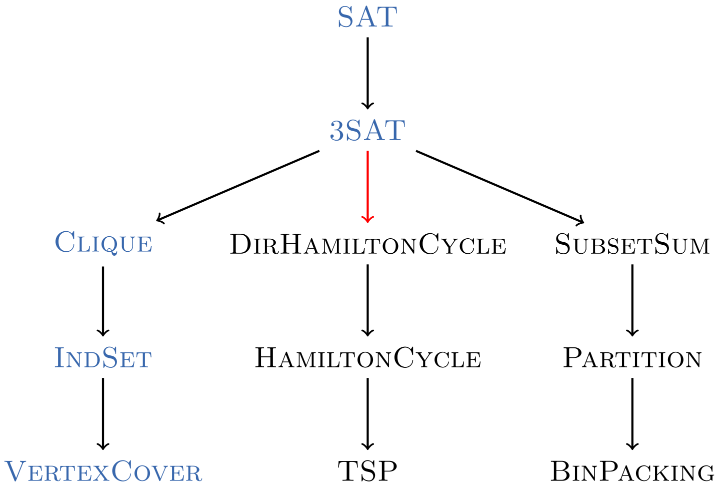


Questions

Questions?

Routing Problems

3SAT $\leq_p$ DirHamiltonCycle



DirHamiltonCycle is NP-Complete (1)

Definition (Reminder: DirHamiltonCycle)

The problem DirHamiltonCycle is defined as follows:

Given: directed graph $G = \langle V, E \rangle$

Question: Does G contain a Hamilton cycle?

DirHamiltonCycle is NP-Complete (1)

Definition (Reminder: DirHamiltonCycle)

The problem DirHamiltonCycle is defined as follows:

Given: directed graph $G = \langle V, E \rangle$

Question: Does G contain a Hamilton cycle?

Theorem

DirHamiltonCycle *is NP-complete.*

DirHamiltonCycle is NP-Complete (2)

Proof.

DirHamiltonCycle $\in$ NP: guess and check.

DirHamiltonCycle is NP-Complete (2)

Proof.

DirHamiltonCycle $\in$ NP: guess and check.

DirHamiltonCycle is NP-hard:
We show 3SAT $\leq_p$ DirHamiltonCycle.

DirHamiltonCycle is NP-Complete (2)

Proof.

DirHamiltonCycle ∈ NP: guess and check.

DirHamiltonCycle is NP-hard:
We show 3SAT $\leq_p$ DirHamiltonCycle.

- We are given a 3-CNF formula φ where each clause contains exactly three literals and no clause contains duplicated literals.
- We must, in polynomial time, construct
  a directed graph $G = \langle V, E \rangle$ such that:
  G contains a Hamilton cycle iff φ is satisfiable.
- construction of $\langle V, E \rangle$ on the following slides

...

DirHamiltonCycle is NP-Complete (3)

Proof (continued).

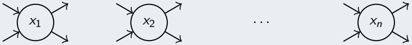
- Let $X_1, \dots, X_n$ be the propositional variables in φ.
- Let $c_1, \dots, c_m$ be the clauses of φ with $c_i = (l_{i1} \lor l_{i2} \lor l_{i3})$.
- Construct a graph with $6m + n$ vertices (described on the following slides).

...

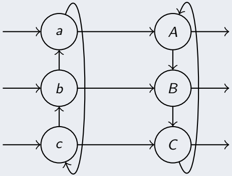
DirHamiltonCycle is NP-Complete (4)

Proof (continued).

- For every variable X_i, add vertex x_i with 2 incoming and 2 outgoing edges:



- For every clause c_j, add the subgraph C_j with 6 vertices:



- We describe later how to connect these parts.

...

DirHamiltonCycle is NP-Complete (5)

Proof (continued).

Let π be a Hamilton cycle of the total graph.

- Whenever π enters subgraph C_j from one of its "entrances", it must leave via the corresponding "exit":
  $(a \longrightarrow A,\ b \longrightarrow B,\ c \longrightarrow C)$.
  Otherwise, π cannot be a Hamilton cycle.
- Hamilton cycles can behave in the following ways with regard to C_j:
  - π passes through C_j once (from any entrance)
  - π passes through C_j twice (from any two entrances)
  - π passes through C_j three times (once from every entrance)

...

DIRHAMILTONCYCLE is NP-Complete (6)

Proof (continued).

Connect the "open ends" in the graph as follows:

- Identify entrances/exits of the clause subgraph C_j with the three literals in clause c_j.
- One exit of x_i is positive, the other one is negative.
- For the positive exit, determine the clauses in which the positive literal X_i occurs:
  - Connect the positive exit of x_i with the X_i-entrance of the first such clause graph.
  - Connect the X_i-exit of this clause graph with the X_i-entrance of the second such clause graph, and so on.
  - Connect the X_i-exit of the last such clause graph with the positive entrance of x_{i+1} (or x_1 if $i = n$).
- analogously for the negative exit of x_i and the literal $\neg X_i$

...

DirHamiltonCycle is NP-Complete (7)

Proof (continued).

The construction is polynomial and is a reduction:

(⇒): construct a Hamilton cycle from a satisfying assignment

- Given a satisfying assignment $\mathcal{I}$, construct a Hamilton cycle that leaves x_i through the positive exit if $\mathcal{I}(X_i)$ is true and by the negative exit if $\mathcal{I}(X_i)$ is false.
- Afterwards, we visit all C_j-subgraphs for clauses that are satisfied by this literal.
- In total, we visit each C_j-subgraph 1–3 times.

...

DirHamiltonCycle is NP-Complete (8)

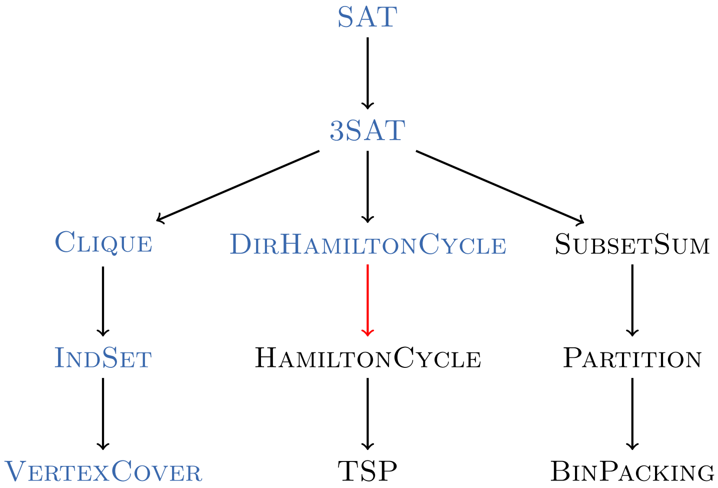
Proof (continued).

(⇐): construct a satisfying assignment from a Hamilton cycle

- A Hamilton cycle visits every vertex x_i and leaves it by the positive or negative exit.
- Map X_i to true or false depending on which exit is used to leave x_i.
- Because the cycle must traverse each C_j-subgraph at least once (otherwise it is not a Hamilton cycle), this results in a satisfying assignment. (Details omitted.)

□

DirHamiltonCycle $\leq_p$ HamiltonCycle



HamiltonCycle is NP-Complete (1)

Definition (Reminder: HamiltonCycle)

The problem HamiltonCycle is defined as follows:

Given: undirected graph $G = \langle V, E \rangle$

Question: Does G contain a Hamilton cycle?

HamiltonCycle is NP-Complete (1)

Definition (Reminder: HamiltonCycle)

The problem HamiltonCycle is defined as follows:

Given: undirected graph $G = \langle V, E \rangle$

Question: Does G contain a Hamilton cycle?

Theorem

HamiltonCycle *is NP-complete.*

HamiltonCycle is NP-Complete (2)

Proof sketch.

HamiltonCycle $\in$ NP: guess and check.

HamiltonCycle is NP-Complete (2)

Proof sketch.

HamiltonCycle $\in$ NP: guess and check.

HamiltonCycle is NP-hard: We show DirHamiltonCycle $\leq_p$ HamiltonCycle.

HamiltonCycle is NP-Complete (2)

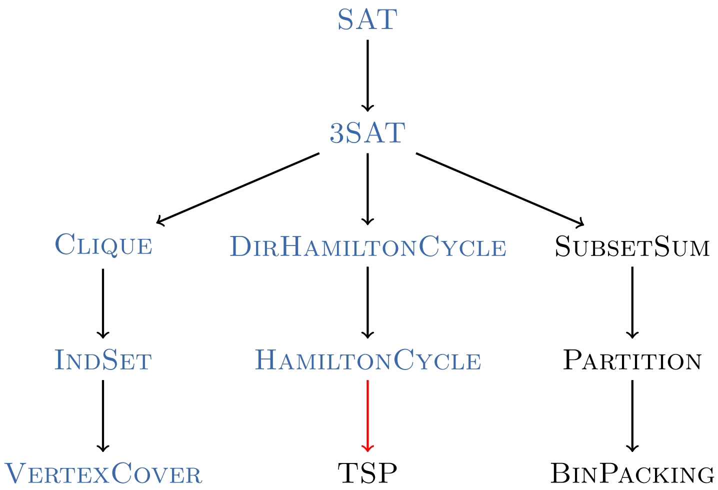
Proof sketch.

HamiltonCycle $\in$ NP: guess and check.

HamiltonCycle is NP-hard: We show DirHamiltonCycle $\leq_p$ HamiltonCycle.

Basic building block of the reduction:

HamiltonCycle $\leq_p$ TSP



TSP is NP-Complete (1)

Definition (Reminder: TSP)

TSP (traveling salesperson problem) is the following decision problem:

- Given: finite set $S \neq \emptyset$ of cities, symmetric cost function $cost : S \times S \to \mathbb{N}_0$, cost bound $K \in \mathbb{N}_0$
- Question: Is there a tour with total cost at most K, i. e., a permutation $\langle s_1, \dots, s_n \rangle$ of the cities with $\sum_{i=1}^{n-1} cost(s_i, s_{i+1}) + cost(s_n, s_1) \leq K$?

German: Problem der/des Handlungsreisenden

Theorem

TSP *is NP-complete.*

TSP is NP-Complete (2)

Proof.

TSP $\in$ NP: guess and check.

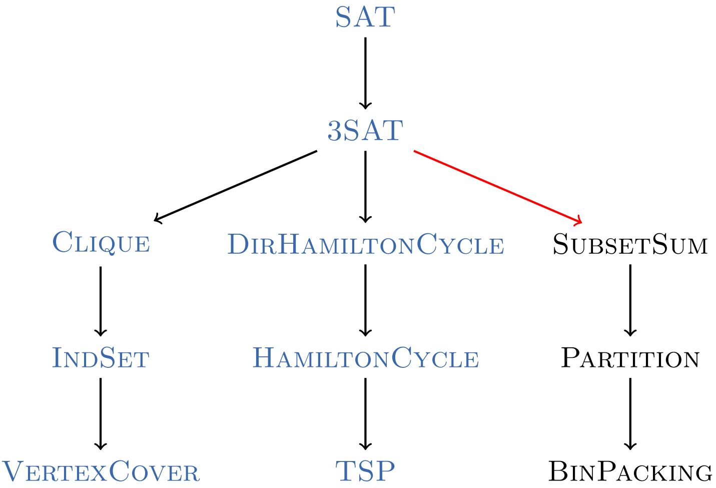
TSP is NP-hard: We showed HamiltonCycle $\leq_p$ TSP in Chapter E2. □

Questions

Questions?

Packing Problems

3SAT $\leq_p$ SubsetSum



SubsetSum is NP-Complete (1)

Definition (SubsetSum)

The problem SubsetSum is defined as follows:

Given: numbers $a_1, \dots, a_k \in \mathbb{N}_0$ and $b \in \mathbb{N}_0$

Question: Is there a subset $J \subseteq \{1, \dots, k\}$ with $\sum_{i \in J} a_i = b$?

Theorem

SubsetSum *is NP-complete.*

SubsetSum is NP-Complete (2)

Proof.

SubsetSum $\in$ NP: guess and check.

SubsetSum is NP-hard: We show $3\mathrm{SAT} \leq_p$ SubsetSum.

SubsetSum is NP-Complete (2)

Proof.

SubsetSum $\in$ NP: guess and check.

SubsetSum is NP-hard: We show $\text{3SAT} \leq_{\mathrm{p}} \text{SubsetSum}$.

Given a 3-CNF formula φ, we compute a SubsetSum instance that has a solution iff φ is satisfiable.

SubsetSum is NP-Complete (2)

Proof.

SubsetSum $\in$ NP: guess and check.

SubsetSum is NP-hard: We show $\mathrm{3SAT} \leq_{\mathsf{p}}$ SubsetSum.

Given a 3-CNF formula φ, we compute a SubsetSum instance that has a solution iff φ is satisfiable.

We can assume that all clauses have exactly three literals and that the literals in each clause are unique.

Let m be the number of clauses in φ,
and let n be the number of variables.

Number the propositional variables in φ in any way,
so that it is possible to refer to "the i-th variable". ...

SubsetSum is NP-Complete (3)

Proof (continued).

The target number of the SubsetSum instance is
$\sum_{i=1}^{n} 10^{i-1} + \sum_{i=1}^{m} 4 \cdot 10^{i+n-1}$
(in decimal digits: m 4s followed by n 1s).

SubsetSum is NP-Complete (3)

Proof (continued).

The target number of the SubsetSum instance is
$\sum_{i=1}^{n} 10^{i-1} + \sum_{i=1}^{m} 4 \cdot 10^{i+n-1}$
(in decimal digits: m 4s followed by n 1s).

The numbers to select from are:

- one number for each literal (X or $\neg X$):
  if the literal belongs to the j-th variable and occurs (exactly) in the k clauses $i_1, \dots, i_k$, its literal number is
  $10^{j-1} + 10^{i_1+n-1} + \dots + 10^{i_k+n-1}$.
- for each clause, two padding numbers:
  10^{i+n-1} and $2 \cdot 10^{i+n-1}$ for all $i \in \{1, \dots, m\}$.

This SubsetSum instance can be produced in polynomial time.

...

SubsetSum is NP-Complete (4)

Proof (continued).

Observations:

- With these numbers, no carry occurs in any subset sum. Hence, to match the target, all individual digits must match.
- For $i \in \{1, \dots, n\}$, refer to the i-th digit (from the right) as the i-th variable digit.
- For $i \in \{1, \dots, m\}$, refer to the $(n + i)$-th digit (from the right) as the i-th clause digit.
- Consider the i-th variable digit. Its target value is 1, and only the two literal numbers for this variable contribute to it.
- Hence, for each variable X, a solution must contain either the literal number for X or for $\neg X$, but not for both.

...

SubsetSum is NP-Complete (5)

Proof (continued).

- Call a selection of literal numbers that makes the variable digits add up a candidate.
- Associate each candidate with the truth assignment that satisfies exactly the literals in the selected literal numbers.
- This produces a 1:1 correspondence between candidates and truth assignments.
- We now show: a given candidate gives rise to a solution iff it corresponds to a satisfying truth assignment.
- This then shows that the SubsetSum instance is solvable iff φ is satisfiable, completing the proof.

...

SubsetSum is NP-Complete (6)

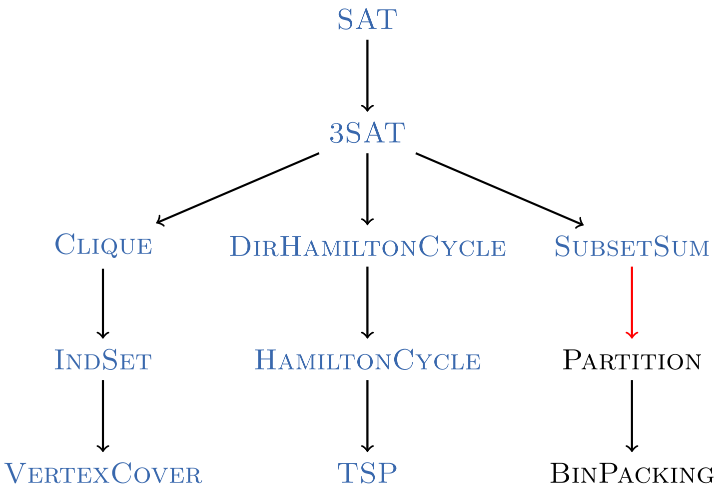
Proof (continued).

Consider a candidate and its corresponding truth assignment.

- Each chosen literal number contributes 1 to the clause digit of each clause satisfied by this literal.
- Satisfying assignments satisfy 1–3 literals in every clause. By using one or both of the padding numbers for each clause digit, all clause digits can be brought to their target value of 4, solving the SubsetSum instance.
- For unsatisfying assignments, there is at least one clause with 0 satisfied literals. It is then not possible to extend the candidate to a SubsetSum solution because the target value of 4 cannot be reached for the corresponding clause digit.

□

SubsetSum $\leq_p$ Partition



PARTITION is NP-Complete (1)

Definition (PARTITION)

The problem PARTITION is defined as follows:

Given: numbers $a_1, \ldots, a_k \in \mathbb{N}_0$

Question: Is there a subset $J \subseteq \{1, \ldots, k\}$ with $\sum_{i \in J} a_i = \sum_{i \in \{1,\ldots,k\} \setminus J} a_i$?

Theorem

PARTITION *is NP-complete.*

PARTITION is NP-Complete (2)

Proof.

PARTITION ∈ NP: guess and check.

PARTITION is NP-Complete (2)

Proof.

PARTITION $\in$ NP: guess and check.

PARTITION is NP-hard: We show SUBSETSUM $\leq_{\mathrm{p}}$ PARTITION.
We are given a SUBSETSUM instance with numbers $a_1, \ldots, a_k$ and target size b. Let $M := \sum_{i=1}^{k} a_i$.

Partition is NP-Complete (2)

Proof.

Partition $\in$ NP: guess and check.

Partition is NP-hard: We show SubsetSum $\leq_p$ Partition.
We are given a SubsetSum instance with numbers $a_1, \ldots, a_k$ and target size b. Let $M := \sum_{i=1}^{k} a_i$.
Construct the Partition instance $a_1, \ldots, a_k, M + 1, 2b + 1$ (can obviously be computed in polynomial time).

PARTITION is NP-Complete (2)

Proof.

PARTITION $\in$ NP: guess and check.

PARTITION is NP-hard: We show SUBSETSUM $\leq_p$ PARTITION.
We are given a SUBSETSUM instance with numbers $a_1, \dots, a_k$ and target size b. Let $M := \sum_{i=1}^{k} a_i$.

Construct the PARTITION instance $a_1, \dots, a_k, M + 1, 2b + 1$ (can obviously be computed in polynomial time).

Observation: the sum of these numbers is
$M + (M + 1) + (2b + 1) = 2M + 2b + 2$
$\leadsto$ A solution partitions the numbers into two subsets, each with sum $M + b + 1$.

...

PARTITION is NP-Complete (3)

Proof (continued).

Reduction property:

(⇒): construct PARTITION solution from SUBSETSUM solution

PARTITION is NP-Complete (3)

Proof (continued).

Reduction property:

(⇒): construct PARTITION solution from SUBSETSUM solution

- Let $J \subseteq \{1, \ldots, k\}$ be a SUBSETSUM solution, i. e. $\sum_{i \in J} a_i = b$.

PARTITION is NP-Complete (3)

Proof (continued).

Reduction property:

(⇒): construct PARTITION solution from SUBSETSUM solution

- Let $J \subseteq \{1, \ldots, k\}$ be a SUBSETSUM solution, i. e. $\sum_{i \in J} a_i = b$.
- Then J together with (the index of) $M + 1$ is a PARTITION solution, since $\sum_{i \in J} a_i + (M + 1) = b + M + 1 = M + b + 1$ (and thus the remaining numbers also add up to $M + b + 1$).

...

PARTITION is NP-Complete (4)

Proof (continued).

($\Leftarrow$): construct SUBSETSUM solution from PARTITION solution

PARTITION is NP-Complete (4)

Proof (continued).

($\Leftarrow$): construct SUBSETSUM solution from PARTITION solution

- One of the two parts of the partition contains the number $M + 1$.

PARTITION is NP-Complete (4)

Proof (continued).

($\Leftarrow$): construct SUBSETSUM solution from PARTITION solution

- One of the two parts of the partition contains the number $M + 1$.
- Then the other numbers in this part sum to $(M + b + 1) - (M + 1) = b$.

PARTITION is NP-Complete (4)

Proof (continued).

($\Leftarrow$): construct SUBSETSUM solution from PARTITION solution

- One of the two parts of the partition contains the number $M + 1$.
- Then the other numbers in this part sum to $(M + b + 1) - (M + 1) = b$.

$\rightsquigarrow$ These remaining numbers must have indices from $\{1, \ldots, k\}$, since $M + 1$ is not one of them and $2b + 1$ is too large.

Partition is NP-Complete (4)

Proof (continued).

(⇐): construct SubsetSum solution from Partition solution

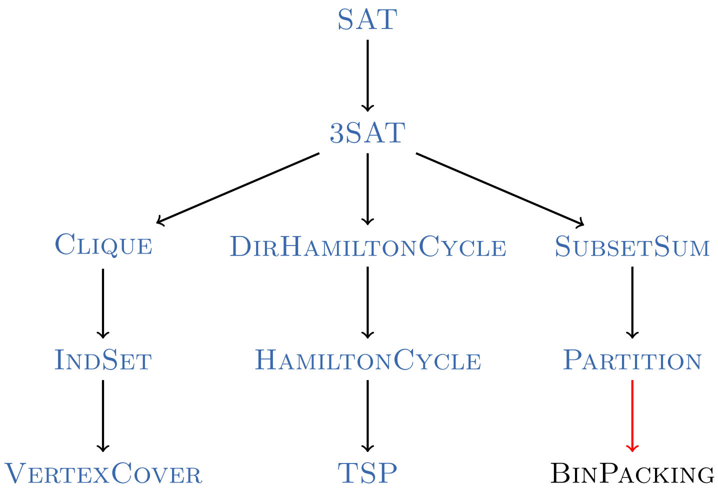
- One of the two parts of the partition contains the number $M + 1$.
- Then the other numbers in this part sum to $(M + b + 1) - (M + 1) = b$.

⇝ These remaining numbers must have indices from $\{1, \dots, k\}$, since $M + 1$ is not one of them and $2b + 1$ is too large.

⇝ These numbers form a SubsetSum solution.

□

PARTITION $\leq_p$ BINPACKING



BinPacking is NP-Complete (1)

Definition (BinPacking)

The problem BinPacking is defined as follows:

Given: bin size $b \in \mathbb{N}_0$, number of bins $k \in \mathbb{N}_0$, objects $a_1, \dots, a_n \in \mathbb{N}_0$

Question: Do the objects fit into the bins?
Formally: is there a mapping $f : \{1, \dots, n\} \to \{1, \dots, k\}$ with $\sum_{i \in \{1,\dots,n\} \text{ with } f(i)=j} a_i \leq b$ for all $1 \leq j \leq k$?

Theorem

BinPacking *is NP-complete.*

BinPacking is NP-Complete (2)

Proof.

BinPacking $\in$ NP: guess and check.

BinPacking is NP-Complete (2)

Proof.

BinPacking $\in$ NP: guess and check.

BinPacking is NP-hard: We show Partition $\leq_p$ BinPacking.

BinPacking is NP-Complete (2)

Proof.

BinPacking $\in$ NP: guess and check.

BinPacking is NP-hard: We show Partition $\leq_p$ BinPacking.

Given the Partition input $\langle a_1, \ldots, a_k \rangle$, we compute $M := \sum_{i=1}^{k} a_i$ and generate a BinPacking input with objects of sizes $a_1, \ldots, a_k$ and 2 bins of size $\lfloor \frac{M}{2} \rfloor$.

BinPacking is NP-Complete (2)

Proof.

BinPacking $\in$ NP: guess and check.

BinPacking is NP-hard: We show Partition $\leq_p$ BinPacking.

Given the Partition input $\langle a_1, \dots, a_k \rangle$, we compute $M := \sum_{i=1}^{k} a_i$ and generate a BinPacking input with objects of sizes $a_1, \dots, a_k$ and 2 bins of size $\lfloor \frac{M}{2} \rfloor$.

This can easily be computed in polynomial time, and clearly $a_1, \dots, a_k$ can be partitioned into two groups of the same size iff this bin packing instance is solvable. $\square$

Questions

Questions?

Conclusion

. . . and Many More

Further examples of NP-complete problems:

- 3-COLORING: can the vertices of a graph be colored with three colors in such a way that neighboring vertices always have different colors?
- MINESWEEPERCONSISTENCY: Is a given cell in a given Minesweeper configuration safe?
- GENERALIZEDFREECELL: Is a given generalized FreeCell tableau (i. e., one with potentially more than 52 cards) solvable?
- . . . and many, many more

Chapter Summary

- In this chapter we showed NP-completeness of further problems:
  - three classical routing problems: DirHamiltonCycle, HamiltonCycle, TSP
  - three classical packing problems: SubsetSum, Partition, BinPacking

Complexity Theory Summary

- Complexity theory investigates which problems are "easy" to solve and which ones are "hard".
- two important problem classes:
  - P: problems that are solvable in polynomial time by "normal" computation mechanisms
  - NP: problems that are solvable in polynomial time with the help of nondeterminism
- We know that $\mathrm{P} \subseteq \mathrm{NP}$, but we do not know whether $\mathrm{P} = \mathrm{NP}$.
- Many practically relevant problems are NP-complete:
  - They belong to NP.
  - All problems in NP can be reduced to them.
- If there is an efficient algorithm for one NP-complete problem, then there are efficient algorithms for all problems in NP.