Chapter 34: NP-Completeness
About this Tutorial

• What is NP?
  • How to check if a problem is in NP?

• Cook-Levin Theorem
  • Showing one of the most difficult problems in NP

• Problem Reduction
  • Finding other most difficult problems
Polynomial time algorithm

- **Polynomial time algorithms**: inputs of size $n$, worst-case running time is $O(n^k)$.
- **Exponential time**: $O(2^n), O(3^n), O(n!), ...$
- It is natural to wonder whether all problems can be solved in polynomial time.
- The answer is no. For example, the "Halting Problem," cannot be solved by any computer no matter how much time we allow.
Tractable vs. Intractable

- Shortest vs. Longest simple paths
- Euler tour vs. Hamiltonian cycle
- 2-CNF (Conjunctive Normal Form) satisfiability vs. 3-CNF satisfiability

For example: a 2-CNF satisfiability problem: 
\((a \lor \neg b) \land (\neg a \lor c) \land (\neg b \lor \neg a)\)

Ans: \(a = 1, b = 0, c = 1\)
Decision Problems

- When we receive a problem, the first thing concern is: whether the problem has a solution or not
- E.g., Peter gives us a map $G = (V, E)$, and he asks us if there is a path from $A$ to $B$ whose length is at most 100
- E.g., Your sister gives you a number, say $1111111111111111111$ (19 one’s), and asks you if this number is a prime
Decision Problems

• The problems in the previous page is called decision problems, because the answer is either YES or NO.

• Some decision problems can be solved efficiently, using time polynomial to the size of the input (what is input size?)

• We use $P$ to denote the set of all these polynomial-time solvable problems.
Decision Problems

E.g., For Peter’s problem, there is an $O(V \log V + E)$-time algorithm that finds the shortest path from $A$ to $B$;

→ we can first apply this algorithm and then give the correct answer

→ Peter’s problem is in $P$

• Can you think of other problems in $P$?
Decision Problems

- Another interesting classification of decision problems is to see if the problem can be verified in time polynomial to the size of the input.
- Precisely, for such a decision problem, whenever it has an answer \textbf{YES}, we can:
  1. Ask for a short proof, and
     /* short means: polynomial in size of input */
  2. Be able to verify the answer is \textbf{YES}.
Decision Problems

e.g., In Peter’s problem, if there is a path from A to B with length \( \leq 100 \), we can:
1. Ask for the sequence of vertices (with no repetition) in any path from A to B whose length \( \leq 100 \)
2. Check if it is a desired path (in poly-time)

\[ \rightarrow \text{this problem is polynomial-time verifiable} \]
Polynomial-Time Verifiable

More examples:

*Given a graph $G = (V,E)$, does the graph contain a Hamiltonian path?*

*Is a given integer $x$ a composite number?*

*Given a set of numbers, can be divide them into two groups such that their sum are the same?*
Now, imagine that we have a super-smart computer, such that for each decision problem given to it, it has the ability to guess a short proof (if there is one).

With the help of this powerful computer, all polynomial-time verifiable problems can be solved in polynomial time (how?)
The Class P and NP

- **NP** denote the set of polynomial-time verifiable problems
  - **N** stands for non-deterministic guessing power of our computer
  - **P** stands for polynomial-time “verifiable”
- **NP**: set of problems can be solved in polynomial time with non-deterministic Turing machine
- **P** denote the set of problems that are polynomial-time solvable
P and NP

• We can show that a problem is in $P$ implies that it is in $NP$ (why?)

• Because if a problem is in $P$, and if its answer is $YES$, then there must be an algorithm that runs in polynomial-time to conclude $YES$ ...

• Then, the execution steps of this algorithm can be used as a “short” proof
P and NP

• On the other hand, after many people’s efforts, some problems in \( \text{NP} \) (e.g., finding a Hamiltonian path) do not have a polynomial-time algorithm yet ...

• Question: Does that mean these problems are not in \( \text{P} \) ??

• The question whether \( \text{P} = \text{NP} \) is still open

Clay Mathematics Institute (CMI) offers US$ 1 million for anyone who can answer this ...
All decision problems

NP

NPC

P

The halting problem and Presburger Arithmetic are in here

NP = P ?
P and NP

• So, the current status is:
  1. If a problem is in \( P \), then it is in \( NP \)
  2. If a problem is in \( NP \), it may be in \( P \)

• In the early 1970s, Stephen Cook and Leonid Levin (separately) discovered that:
  a problem in \( NP \), called \( SAT \), is very mysterious …
Cook-Levin Theorem

• If SAT is in $P$, then every problems in $NP$ are also in $P$
  • i.e., if SAT is in $P$, then $P = NP$

// Can Cook or Levin claim the money from CMI yet?

• Intuitively, SAT must be one of the most difficult problems in $NP$
  • We call SAT an $NP$-complete problem (most difficult in $NP$)
Satisfiable Problem

• The SAT problem asks:
  • Given a Boolean formula $F$, such as
    $$F = (x \lor y \lor \neg z) \land (\neg y \lor z) \land (\neg x)$$
    is it possible to assign True/False to each variable, such that the overall value of $F$ is true?

Remark: If the answer is YES, $F$ is a satisfiable, and so it is how the name SAT is from
Other NP-Complete Problems

• The proofs made by Cook and Levin is a bit complicated, because intuitively they need to show that no problems in $\text{NP}$ can be more difficult than $\text{SAT}$

• However, since Cook and Levin, many people show that many other problems in $\text{NP}$ are shown to be $\text{NP}$-complete

• How come many people can think of complicated proofs suddenly ??
Problem Reduction

• How these new problems are shown to be NP-complete rely on a new technique, called reduction (problem transformation)

• Basic Idea:
  • Suppose we have two problems, A and B
  • We know that A is very difficult
  • However, we know if we can solve B, then we can solve A
  • What can we conclude ??
Problem Reduction

- e.g., $A = \text{Finding median}$, $B = \text{Sorting}$
- We can solve $A$ if we know how to solve $B$
  $\implies$ sorting is as hard as finding median

- eg., $A = \text{Topological Sort}$, $B = \text{DFS}$
- We can solve $A$ if we know how to solve $B$
  $\implies$ DFS is as hard as topological sort
Problem Reduction

• Now, consider
  \[ A = \text{an NP-complete problem (e.g., SAT)} \]
  \[ B = \text{another problem in NP} \]

• Suppose that we can show that:
  1. we can transform a problem of \( A \) into a problem of \( B \), using polynomial time
  2. We can answer \( A \) if we can answer \( B \)

  \[ \Rightarrow \text{Then we can conclude } B \text{ is NP-complete} \]
  (Can you see why??)
Problem Reduction

• All satisfiability problem can be reduced to 3-SAT problem in polynomial time.
• For example,
  ✓ $(x_1 \lor x_2) \Rightarrow (x_1 \lor x_2 \lor y_1) \land (x_1 \lor x_2 \lor \neg y_1)$
  ✓ $\neg x_3 \Rightarrow (\neg x_3 \lor y_1 \lor y_2) \land (\neg x_3 \lor \neg y_1 \lor y_2)$
    $\land (\neg x_3 \lor y_1 \lor \neg y_2) \land (\neg x_3 \lor \neg y_1 \lor \neg y_2)$
  ✓ $(x_1 \lor x_2 \lor x_3 \lor x_4) \Rightarrow (x_1 \lor x_2 \lor y_1) \land (x_3 \lor x_4 \lor \neg y_1)$
• Since 3-SAT is in NP, 3-SAT is a NP-Complete problem
NP-Complete & NP-Hard

- **NP-Complete**: a problem A is in NPC iff (i) A is in NP, and (ii) any problem in NPC can be reduced to it in $O(n^k)$ time.

- If any problem in NPC can be solved in $O(n^k)$ time, then $P=NP$. It is believed (but not proved) that $P \neq NP$.

- **NP-Hard**: a problem A is in NPH iff a problem in NPC can be reduced to A in $O(n^k)$ time.
NP-hard, NP, Np-Complete and P

If NP ≠ P

If NP = P

P = NP = NP-Complete
Example

• Let us define two problems as follows:

  • The \textbf{CLIQUE} problem:
    \hspace{1cm} \textit{Given a graph } G = (V,E), \textit{and an integer } k, \textit{does the graph contain a complete graph with at least } k \textit{ vertices}

  • The \textbf{IND-SET} problem:
    \hspace{1cm} \textit{Given a graph } G = (V,E), \textit{and an integer } k, \textit{does the graph contain } k \textit{ vertices such that there is no edge in between them?}
Example

• Questions:
  1. Are both problems decision problems?
  2. Are both problems in \( \text{NP} \)?

• In fact, \( \text{CLIQUE} \) is \( \text{NP} \)-complete

  • Can we use reduction to show that \( \text{IND-SET} \) is also \( \text{NP} \)-complete?

  [ transform which problem to which?? ]
Examples

• 3-CNF satisfiability problem (3SAT):
  \((a \lor b \lor c) \land (a \lor d \lor e) \land (b \lor f \lor a)\)

• The subset-sum (partition) problem: partition a set of (real) numbers into two subsets of the same sum

• The k-graph coloring problem \((k \geq 3)\)

• The traveling-salesperson problem (TSP)

• The vertex-cover problem

• The independent set problem
True or False

• If a problem is NPC, then it can not be solved by any polynomial time algorithm in worst cases.
• If a problem is NPC, then we have not found any polynomial time algorithm to solve it in worst cases.
• If a problem is NPC, then it is unlikely that a polynomial time algorithm can be found in the future to solve it in worst cases.
True or False

• If we can prove that the lower bound of an NPC problem is exponential, then we have proved that $\text{NP} \neq \text{P}$.

• Any NP-hard problem can be solved in polynomial time if there is an algorithm that can solve the satisfiability problem in polynomial time.
Homework

• Practice at home: 34.1-4, 34.1-5, 34.4-7
Quiz

• Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How fast can you make Prim’s algorithm run?

• How to solve the single-source shortest paths problem in directed acyclic graphs (DAGs)?
Test

• The NP problems consist of only decision problems (True or False?)

• If the worst case time-complexity of an algorithm is $O(2^n)$, it is an exponential time algorithm (True or False?)