CS 5319
Advanced Discrete Structure

Lecture 17:
Introduction to NP-Completeness

Outline

- What is NP ?
  - How to check if a problem is in NP ?
- Cook-Levin Theorem
  - One of the most difficult problem in NP
- Problem Reduction
  - Finding other most difficult problems

What is NP ?

Decision Problems

- When we receive a problem, our first concern is: whether the problem has a solution or not
- Ex : 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
- Ex : 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 a decision problem, because the answer is either YES or NO
- Some decision problems can be solved efficiently, using time polynomial to the size of the input
  - We use ***P*** to denote the set of all these polynomial-time solvable problems

Decision Problems

Ex: For the previous graph 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

→ the 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 input
- Precisely, for such a decision problem, whenever it has an answer 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 YES

Decision Problems

Ex :

In the previous graph problem, if there is a path from A to B with length ≤ 100, we can :

1. Ask for the sequence of vertices (with no repetition) in any path from A to B whose length ≤ 100
2. Check if it is a desired path (in polynomial time)

➔ this problem is polynomial-time verifiable

Polynomial-Time Verifiable

More examples:

- Given a graph $G = (V, E)$, does the graph contain a Hamiltonian path ?
- Given a set of numbers, can we divide them into two groups such that the sum of each group are the same ?
- Given an integer x, is x a composite number ?

Polynomial-Time Verifiable

- Imagine we have a smart computer, such that
  - for each decision problem given to it, it can guess a correct 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 ***NP***

- Many problems are polynomial-time verifiable
- Because of this, we use ***NP*** to denote the set of polynomial-time verifiable problems
  - ***N*** stands for (non-deterministic) guessing power of our computer
  - ***P*** stands for 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 …
  - The execution steps of this algorithm (and the algorithm itself) *is* a "short" proof

P and ***NP***

- On the other hand, after many people's efforts, some problems in ***NP*** (e.g., finding a Hamiltonian path) do not have a polynomial-time algorithm yet …
- Question: Does that mean these problems are not in ***P*** ??
- The question " Is ***P*** = ***NP*** ? " is still open

Clay Mathematics Institute (CMI) offers US$ 1 million for anyone who can answer this …

Cook-Levin Theorem

Current Status

- 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:

  an ***NP*** problem called SAT is very mysterious

Cook-Levin Theorem

Theorem (Cook-Levin) :

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

The SAT Problem

- The SAT problem asks:
  - Given a Boolean formula F, such as

$$F = (x \vee y \vee \neg z) \wedge (\neg y \vee z) \wedge (\neg x)$$

is it possible to assign true/false to each variable, such that the value of F is true ?

Remark: If the answer is YES, F is a satisfiable, and so it is how the name SAT comes from

Other ***NP***-complete Problems

- The proofs made by Cook and Levin is a bit complicated, because intuitively we need to show that no problems in ***NP*** can be more difficult than SAT
- However, since Cook and Levin, many other problems in ***NP*** are shown to be ***NP***-complete
  - How come people can think of many complicated proofs suddenly ??

Problem Reduction

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
  - Also, if we can solve B, we can solve A
  - What can we conclude ??

Problem Reduction

- Now, consider

  A = an ***NP***-complete problem (e.g., SAT)

  B = 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. if we can answer B, we can answer A

➔ Then we can conclude B is ***NP***-complete

Example

- Let us define two problems as follows :
- The CLIQUE problem:

  Given a graph G = (V,E), and an integer k, does *G* contain a complete graph with k vertices ?

- The IND-SET problem:

  Given a graph G = (V,E), and an integer k, does G contain k vertices such that there is no edge in between them ?

Example

Questions:

1. Are both problems decision problems ?
2. Are both problems in ***NP*** ?

- In fact, CLIQUE is ***NP***-complete
  - Can we use reduction to show that IND-SET is also ***NP***-complete ?

    [transform which problem to which ??]