15 Nondeterminism and NP

15.1 A Larger Class than P

In Section 13, we saw that it can be useful to have a class of problems that is a little larger than the class that you are interested in. For example, that allows you to identify problems that are among the most difficult problems in the larger class. If class C is a subset of class D and X is one of the most difficult problems in the larger class D, you would expect X not to be in the smaller class C.

In this section, we introduce a class NP that (we hope) is larger than P.

15.2 Mersenne's Conjecture

In 1644, Marin Mersenne made what came to be known as Mersenne's conjecture: $2^n - 1$ is prime for n = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127 and 257. and is composite for all other positive integers $n \leq 257$.

Mersenne's conjecture stood until 1903 when Frank Cole made a presentation that put it to rest. The presentation was quite short. By starting with 2, successively doubling and finally subtracting 1, Cole showed that

$$2^{67} - 1 = 147,573,952,589,676,412,927.$$

Then he wrote down two numbers and multiplied them together.

$$\begin{array}{lr} & 761{,}838{,}257{,}287 \\ \times & 193{,}707{,}721 \\ \hline & 147{,}573{,}952{,}589{,}676{,}412{,}927 \end{array}$$

That was all it took to convince Cole's audience that Mersenne's conjecture was mistaken. (Other mistakes in it were discovered later.)

But where did Cole get the factors of $2^{67} - 1$? He said that it took "three years of sundays" to find them.

In idealized form, our system of justice is supposed to work as follows. First, the police gather evidence. Then, the prosecutor presents the case to a jury. Finally, the jury rules on whether the evidence is convincing. If the jury does not find the evidence convincing, the jurors are not required to go out and find new evidence. The case is over.

Frank Cole played the role of police and prosecutor and his audience played the role of jury. It can take much less time to present a case than it does to find the evidence.

15.3 Evidence Checkers

We can break down testing whether string x is in language A into two parts: finding evidence and checking the evidence.

Definition 15.1. A *polynomial-time evidence checker* for language A is a program check(e, x) where there exists a positive integer k so that

1. check(e, x) runs in polynomal time (in the length of ordered pair (e, x)).
2. If $x \in A$ then there is a string e (the evidence) where $|e| \leq |x|^k$ and check$(e, x) = 1$. That is, members of A have short, easy to check evidence that they are members of A. (The jury correctly recognizes convincing evidence.)
3. If $x \notin A$ then there does not exist any string e so that check$(e, x) = 1$. That is, the evidence checker cannot be fooled into believing that $x \in A$ when in fact $x \notin A$. (The jury does not convict on bad evidence.)

There is an important asymmetry in evidence checkers. If $x \in A$, then there must be checkable evidence that $x \in A$. But if $x \notin A$, no evidence is required showing that $x \notin A$. All that is required is a lack of evidence that $x \in A$.

15.4 Definition of NP

Definition 15.2. *NP* is the class of all decision problems that have polynomial-time evidence checkers.

For example, an integer x is *composite* if $x > 1$ and x is not prime. The smallest composite number is 4 ($= 2 \cdot 2$). Define

$$\text{COMPOSITE} = \{x \in \mathcal{N} \mid x \text{ is composite}\}.$$

It is easy to see that COMPOSITE is in NP. (Frank Cole showed how.) The following is a polynomial-time evidence checker for COMPOSITE where the evidence e should be a factor of x and the checker verifies that.

```
"{composite(e, x):
  If 1 < e < x and x mod e == 0
    return 1
  else
    return 0
}"
```

A simpler way to present an evidence checker is to list (1) the input, (2) the evidence and (3) the conditions that needs to be satisfied for the evidence to be convincing.

Evidence checker for COMPOSITE	
Input	Positive integer x
Evidence	Positive integer e
Requirement	$1 < e < x$ and $x \bmod e = 0$

15.5 Examples of Problems In NP

15.5.1 Is a given propositional formula satisfiable?

Definition 15.3. A propositional formula ϕ is *satisfiable* if there exists a truth-value assignment for the variables in ϕ that makes ϕ true.

Definition 15.4. The *Satisfiability Problem for Propositional Logic* (SATPL) is the following decision problem.

Input. A propositional formula ϕ.
Question. Is ϕ satisfiable?

Theorem 15.5. SATPL is in NP.

Proof. All we need is a polynomial-time evidence checker for SATPL. If you think about a truth-table for ϕ, you only need to look at one row to determine that ϕ is satisfiable.

Evidence checker for SATPL	
Input.	Propositional formula ϕ
Evidence.	Truth value assignment a
Requirement.	$(a \dashv \phi)$ is true.

$\diamondsuit$

15.5.2 Does a Graph Have a Small Vertex Cover?

Definition 15.6. Suppose that $G = (V, E)$ is a simple graph. A *vertex cover* of G is a subset $C \subseteq V$ such that, for every edge $\{u, v\} \in E$, $C \cap \{u, v\} \neq \{\}$. That is, every edge must be incident on at least one member of the vertex cover C.

Definition 15.7. The *Vertex Cover Problem* (VCP) is the following decision problem.

Input. A simple graph G and a positive integer k.
Question. Does there exist a vertex cover C of G where $|C| \leq k$?

Theorem 15.8. VCP $\in$ NP.

Proof. Decision problems in NP are often stated as a question about whether something exists. For example, VCP asks whether a vertex cover of a limited size exists. To find a polynomial-time evidence checker, use the thing whose existence is questioned as the evidence. The following is an evidence checker for VCP.

Evidence checker for VCP	
Input	Simple graph $G = (V, E)$ and positive integer k
Evidence	Set $C \subseteq V$
Requirement	$\|C\| \leq k$ and C is a vertex cover of G.

$\diamondsuit$

15.5.3 Can a List of Integers Be Partitioned Equally?

Definition 15.9. A list of positive integers x_1, x_2, $\ldots$, x_n is *equally partitionable* if there exists an index set $I \subseteq \{1, 2, \ldots n\}$ such that

$$\sum_{i \in I} x_i = \sum_{i \notin I} x_i.$$

For example, suppose the list of integers is $x_1 = 14$, $x_2 = 10$, $x_3 = 5$, $x_4 = 7$, $x_5 = 2$, $x_6 = 4$, $x_7 = 6$. Index set $\{1,\ 6,\ 7\}$ equally partitions that list since $x_1 + x_6 + x_7 = 14 + 4 + 6 = 24$ and $x_2 + x_3 + x_4 + x_5 = 10 + 5 + 7 + 2 = 24$.

Definition 15.10. The Partition Problem (PP) is the following decision problem.

Input. A list x_1, x_2, $\ldots$, x_n of positive integers.
Question. Is x_1, x_2, $\ldots$, x_n equally partitionable?

Theorem 15.11. PP $\in$ NP.

Proof. List x_1, x_2, $\ldots$, x_n is equally partitionable if *there exists* an index set I so that

$$\sum_{i \in I} x_i = \sum_{i \notin I} x_i.$$

That suggests using I as the evidence.

Evidence checker for PP	
Input	List x_1, x_2, $\ldots$, x_n of positive integers
Evidence	Index set $I \subseteq \{1,\ \ldots\ n\}$
Requirement	$\displaystyle\sum_{i \in I} x_i = \sum_{i \notin I} x_i$

$\diamondsuit$

15.6 Every Problem In NP Is Computable

Theorem 15.12. If $A \in \text{NP}$ then A is computable.

Proof. Suppose that $A \in \text{NP}$. Let $c(e, x)$ be a polynomial-time evidence checker for A. By the definition of a polynomial-time evidence checker, there is an integer k so that $x \in A$ if and only if there exists a string e with $|e| \leq |x|^k$ and $c(e, x) = 1$.

An algorithm can decide whether $x \in A$ by computing $c(e, x)$ for every string e where $|e| \leq x^k$, answering yes if any of those yields 1.

♢

15.7 Every Problem In P Is Also In NP

Theorem 15.13. Every language that is in P is also in NP. That is, $\text{P} \subseteq \text{NP}$.

Proof. Suppose that A is a language in P. By definition, that means there is a polynomial-time algorithm $\text{inA}(x)$ where $\text{inA}(x) = 1 \leftrightarrow x \in A$. The following is a polynomial-time evidence checker for A. It does not need the evidence, so it ignores the evidence.

Evidence checker for A	
Input	x
Evidence	any string e
Requirement	$\text{inA}(x) = 1$

♢

15.8 The P = NP Question

Think of a problem in NP as a kind of puzzle. Solving a puzzle requires finding a solution, which amounts to evidence that the puzzle has a solution. Often, the solution for a puzzle in the newspaper or a book is provided, and peeking at the solution is much less time consuming (though less satisfying) than finding the solution yourself.

Theorem 15.13 tells us that $\mathrm{P} \subseteq \mathrm{NP}$. But is $\mathrm{NP} \subseteq \mathrm{P}$? If $\mathrm{NP} \subseteq \mathrm{P}$, then, at least up to a polynomial, peeking at the solution is not helpful; you can just find the solution yourself.

If $\mathrm{NP} \subseteq \mathrm{P}$ then $\mathrm{P} = \mathrm{NP}$. Surprisingly, nobody knows whether $\mathrm{P} = \mathrm{NP}$. It is widely *conjectured* that $\mathrm{P} \neq \mathrm{NP}$, but we have already seen that a conjecture can stand for over 200 years only to be overturned.

prev **next**