

This and other notes are available from <http://www.cs.berkeley.edu/~yozo/cs170.fa05/>

In complexity theory, problems are often assumed to be a YES-NO problem; it takes an input and returns either “YES” or “NO”.

- Class P (Polynomial)

The set of all “problems” that can be solved in polynomial time.
- Class NP (Nondeterministic Polynomial)

The set of all “problems” that has a polynomial time verifier. This means that given some *certificate* (often the solution itself), one can check (in polynomial time) that the answer to the problem is indeed “YES”. Whether $P = NP$ or not is an open problem.
- Class NP-Complete

In some sense its the “hardest” of all NP problems. A problem A is said to be NP-Complete if *all* problems in NP reduces polynomially to A . In another words, all problems in NP can be solved using a solver for A , with polynomial overhead.

 1. Explain why if one can solve a single NP-Complete problem in polynomial, then *every* problem in NP can be solved in polynomial time.
 2. If a problem A can be reduced to B , then what can we say about how “hard” these problem are?
 3. Cook-Levin Theorem

Circuit Satisfiability (CIRCUIT-SAT) is NP-Complete.
 4. How is CIRCUIT-SAT different from circuit evaluation?
 5. Why is CIRCUIT-SAT \in NP?
 6. Show that if a problem A reduces to CIRCUIT-SAT then A is in NP.
 7. Give a plausible explanation why any problem A in NP might reduce to CIRCUIT-SAT by noticing that computers (as implemented today) are just a huge CIRCUIT-SAT problem. There is a much more rigorous proof to Cook-Levin Theorem, as you’ll see in CS 172. There you’ll see a rigorous definition of what a “problem” and a “machine” are.
 8. TRUE or FALSE: If $P = NP$, a problem A in NP is NP-complete if and only if A can be reduced polynomially to CIRCUIT-SAT. Why or why not?
 9. TRUE or FALSE: If $P = NP$, a problem A in NP is NP-complete if and only if CIRCUIT-SAT can be reduced polynomially to A . Why or why not?
- More class hierarchies:

$P \subseteq NP \subseteq PSPACE = NPSpace \subseteq EXPTIME \subseteq EXPSPACE \subseteq DECIDABLE$

PSPACE is the class of all problems requiring polynomial memory space. DECIDABLE is the class of all problems where we can solve the problem in finite time (i.e., gives a YES-NO answer in finite time). We know that $P \neq EXPTIME$, but we don’t know much else.

 1. Why is $P \subseteq NP$?
 2. Why is $NP \subseteq PSPACE$?

- Undecidable problems. Consider the HALTING problem: determine whether a program P terminates on input x . If there is a program H that can decide this, then we can construct the following program D that on input P does the following:

$$D(P) : \text{if } H(P, P) = \text{No then loop forever}$$

What happens when we run $D(D)$? It terminates if and only if it doesn't terminate (Why?). This is absurd; hence such a program H cannot exist. In another words, we cannot possibly write a program that determines whether a program terminates or not.

- Standard way to show a problem A is NP-Complete: a problem A is NP-Complete if
 - (a) A is in NP, and
 - (b) An already known NP-complete problem B reduces to A . Be careful of the direction of the reduction: we already know that A reduces to B . Why?
- NP-Complete Problems
CIRCUIT-SAT, 3-SAT, INDEPENDENT SET, VERTEX COVER, CLIQUE, INTEGER LINEAR PROGRAMMING, TRAVELING SALESMAN PROBLEM, HAMILTONIAN PATH/CYCLE, KNAPSACK, SUBSET-SUM, ...
- TRUE or FALSE: SUBSET-SUM can be reduced to CLIQUE.

More Questions

1. Consider the knapsack problem. Explain why the algorithm presented in class is not considered polynomial time.
2. Show that 3-SAT is NP-Complete.
3. Show that INDEPENDENT SET problem is NP-complete.
4. Show that CLIQUE problem is NP-complete. (CLIQUE problem is to find a clique of size k in an undirected graph G).
5. Show that finding the maximal clique (including the nodes forming the clique) is no harder (in polynomial sense) than the YES-NO clique problem above.
6. TRUE or FALSE: A problem in NP is NP-complete if and only if it can be reduced to INDEPENDENT SET problem.