Tutorial 8

NP-Complete Problems

Decision Problem

- Statement of a decision problem
  - Part 1: instance description defining the input
  - Part 2: question stating the actual yes-or-no question
- A decision problem is a mapping from all possible inputs into the set {*yes*, *no*}

The Class *P*

- A **polynomially bounded algorithm** is one with its worst-case complexity bounded by a polynomial function of the input size.
- A **polynomially bounded problem** is one for which there is a polynomially bounded algorithm.
- **The class *P* is the class of decision problems that are polynomially bounded**.

Nondeterministic Algorithm

Phase 1 Guessing: generating **arbitrarily** "certificate", i.e. proposed solution

```
void nondetA(String input)
    String s=genCertif();
    Boolean CheckOK=verifyA(input,s);
    if (checkOK)
        Output "yes";
    return;
```

The algorithm may behave differently on the same input in different executions: "yes" or "no output".

Phase 2 Verifying: determining if *s* is a valid description of a object for answer, and satisfying the criteria for solution

The Class *NP*

- A **polynomial bounded nondeterministic algorithm** is one for which there is a (fixed) polynomial function p such that for each input of size n for which the answer is *yes* , there is some execution of the algorithm that produces a *yes* output in at most $p(n)$ steps.
- The **class *NP*** is the class of decision problems for which there is a polynomial bounded nondeterministic algorithm.

NP-complete Problems

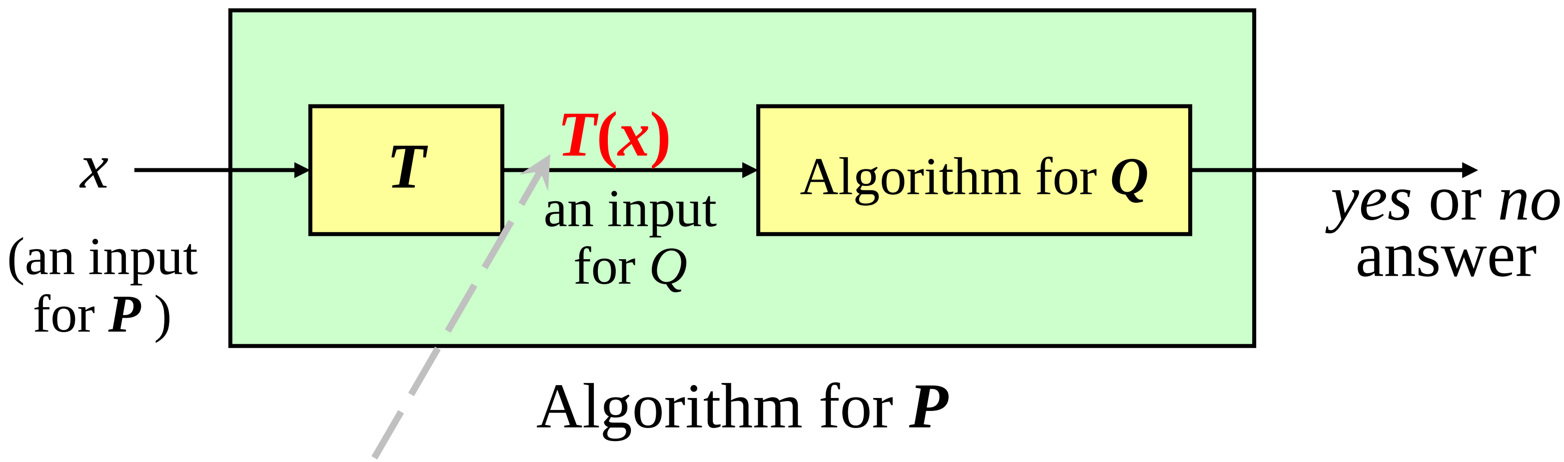
- A problem *Q* is ***NP*-hard** if every problem *P* in *NP* is reducible to *Q*, that is $P \leq_P Q$.

  (which means that *Q* is at least as hard as any problem in *NP*)

- A problem *Q* is ***NP*-complete** if it is in *NP* and is ***NP*-hard**.

  (which means that *Q* is at most as hard as to be solved by a polynomially bounded nondeterministic algorithm)

Polynomial Reduction



$\boldsymbol{P}\,(x) = \mathit{yes} \Leftrightarrow \boldsymbol{Q}\,(T(x)) = \mathit{yes}.$

If x->T(x) is polynomial bounded, we said P is polynimially reducable to Q, denoted as P≤PQ

***(1)If P≤PQ*, then Q is at least as "hard" to solve as P;**

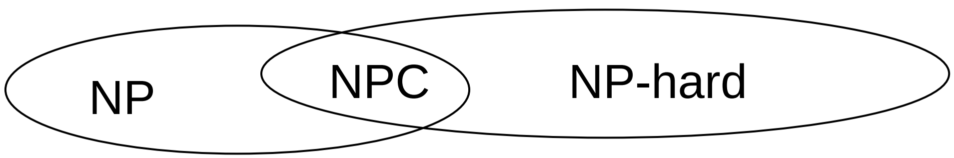
(2) If P≤PQ and Q is in class P, then P is in class P, and the cost of P is bounded by p(n)+q(p(n))

Relation between *P* and *NP*

- $P \subseteq NP$
- $NP \subseteq P$? *Not known yet!*

Relation between *NP* , *NP-hard* and *NPC*

- (1) They are different classes, NP$\neq$NP-hard, as there exists Halt problem, which in NP-hard, but not in NP
- (2)



- (3) If any NP-completed problem is proved in P , then NP =P .
- First Known *NP*-Complete Problem

  Cook' Theorem: SAT$\in$ NPC

Proof of Being in *NP*

- Graph coloring is in *NP*
  - Description of the input and the certificate
  - Properties to be checked for an answer "*yes*"
    - There are *n* colors listed
    - Each c_i is in the range 1,…,*k*
    - Scan the list of edges to see if a conflict exists
  - Proving that each of the above statement can be checked in polynomial time.

Proving *NPC* by Reduction

- The *CNF-SAT* problem is *NP*-complete.
- Prove problem *Q* is *NP*-complete, given a problem *P* known to be *NP*-complete
  - For all $R \in NP$, $R \leq_P P$;
  - **Show $P \leq_P Q$**;
  - By transitivity of reduction, for all $R \in NP$, $R \leq_P Q$;
  - So, *Q* is *NP*-hard;
  - If *Q* is in *NP* as well, then *Q* is *NP*-complete.

Known *NP*-Complete Problem

- Garey & Johnson: *Computer and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979
  - About 300 problems
  - i.e. SAT, Clique, Hamiltonian, Partition, Knapsack …
  - Note: 0-1 knapsack problem is NPC problem, but it can be solved by using dynamic programming in polynomial time, think about why and how? (You can refer to the exercise in CLRS page 384, ex 16.2-2, which solution can be found in the instructor’s manual.) Is that means it in P? No, the DP algorithm is in a pseudo-polynomial time. It is still exponential to its input size. The explanation can be found in the textbook page 558, section 13.2.5.