CS711008Z Algorithm Design and Analysis

Lecture 4. **NP** and intractability (Part II) [1]

Dongbo Bu

Institute of Computing Technology
Chinese Academy of Sciences, Beijing, China

[1]The slides were prepared based on Introduction to algorithms, Algorithm design, Computer and Intractability, and slides by Kevin Wayne with permission.

Outline

- Reduction: understanding the relationship between different problems. $A \leq_P B$ implies "B is harder than A".
- Problem classes: **P**, **NP**, **coNP**, **L**, **NL**, **PSPACE**, **EXP**, etc.
- CIRCUIT SATISFIABILITY is one of the hardest problems in **NP** class.
- **NP-Complete** problems

Complexity classes

- A complexity class of problems is specified by several parameters:
  1. Computation model: multi-string Turing machine;
  2. Computation mode: When do we think a machine accepts its input? deterministic or non-deterministic?
  3. Computation resource: time, space.
  4. Bound: a function f to express how many resource can we use.
- The complexity class is then defined as the set of all languages decided by a multi-string Turing machine M operating in the deterministic/non-deterministic mode, and such that, for input x, M uses at most $f(|x|)$ units of time or space.

(See ppt for description of Turing machine.)

Deterministic versus nondeterministic

- **DTM**: In a deterministic Turing machine, the set of rules prescribes at most one action to be performed for any given situation.
- **NTM**: A non-deterministic Turing machine (NTM), by contrast, may have a set of rules that prescribes more than one actions for a given situation.
- For example, a non-deterministic Turing machine may have both "If you are in state 2 and you see an 'A', change it to a 'B' and move left" and "If you are in state 2 and you see an 'A', change it to a 'C' and move right" in its rule set.

Example: NFA and DFA

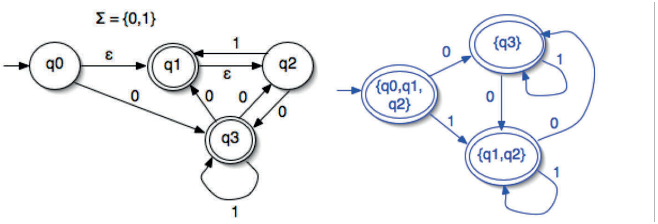
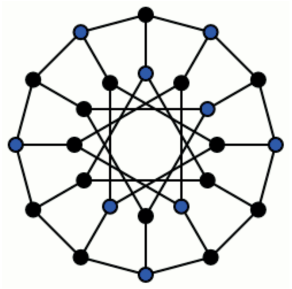


Figure: NFA and DFA

- Perhaps the easiest way to understand determinism and nondeterminism is by looking at NFA and DFA.
- In a DFA, every state has exactly one outgoing arrow for every letter in the alphabet.
- However, the NFA in state 1 has two possible transitions for the letter "b".

DTM vs. NTM: the difference between finding and verifying a solution



- Consider the INDEPENDENT SET problem: does the given graph have an independent set of 9 nodes?
- If your answer is "Yes", you just need to provide a **certificate** having 9 nodes.
- **Certifier:** it is easy to verify whether the certificate is correct, i.e., the given 9 nodes form an independent set for this graph of 24 vertices.
- **Solver:** However, it is not easy to find this independent set

Another example

- Consider the following problem: does the formula $f(x) = x^5 - 3x^4 + 5x^3 - 7x^2 + 11x - 13 = 0$ have a real-number solution?
- If your answer is "`Yes`", you just need to provide a **certificate**, say $x = 0.834\ldots$.
- **Certifier:** it is easy to verify whether the certificate is correct, i.e., $f(x) = 0$.
- **Solver:** however, it is not easy to get a solution.

P class

- **P**: decision problems for which there is a polynomial-time algorithm to **solves** it.
- Here, we say that an algorithm A **solves** problem X if for all instance s of X, $A(s) = \texttt{YES}$ iff s is a YESinstance.
- Time-complexity: A runs in polynomial-time if for **every** instance s, $A(s)$ ends in at most $polynomial(|s|)$ steps.
- Stable Matching problem: $O(n^2)$.

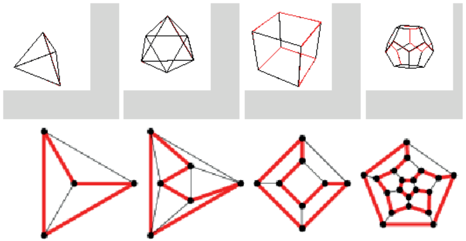
NP class

- **NP**: decision problems for which there exists a polynomial-time **certifier**. [2]
- Here we say that an algorithm $C(s,t)$ **certificates** problem X if for each "YES" instance s, there exists a **certificate** t such that $C(s,t) = \texttt{YES}$, and $|t| = polynomial(|s|)$.
- **Certificate**: an evidence to demonstrate this instance is YES;
- Note: a certifier approach the problem from a **managerial** point of view as follows:
    - It will not actually try to solve the problem directly;
    - Rather, it is willing to efficiently evaluate proposed "proof".

[2] **NP** denotes "non-deterministic polynomial-time". This is just simple but equivalent definition.

Certificate and certifier for HAMILTON CYCLE problem

- Problem: Is there a Hamiltonian cycle in the give graph?
- If your answer is YES, you just need to provide a certificate, i.e. a permutation of n nodes;
- Certifier: checking whether this path forms a cycle;
- Example:
- Certifier: it takes polynomial time to verify the certificate. Thus HAMILTON CYCLE is in **NP** class.



Certificate and certifier for SAT problem

- Problem: Is the given **CNF** satisfiable?
- If your answer is `YES`, you just need to provide a certificate, i.e. an assignment for all n boolean variables;
- Certifier: checking whether each clause is satisfied by this assignment;
- Example:
  - An instance: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$
  - Certificate: $x_1 = \mathtt{TRUE}, x_2 = \mathtt{TRUE}, x_3 = \mathtt{FALSE}$;
  - Certifier: it takes polynomial time to verify the certificate. Thus SAT is in **NP** class.