2. P and NP

- An algorithm is said to be *polynomial bounded* if its worst-case complexity $T(n)$ is bounded by a polynomial function of the input size n:
$$T(n) = O(n^k).$$

  Examples:
  algorithms for LCS, shortest path, MST, ...

- **P** $=$ the class of decision problems that can be **solved** in polynomial time, i.e., they are polynomial bounded

2. P and NP

- **NP** = the class of decision problems that are **verifiable** in polynomial time.

  i.e., if we were given a "certificate" (= a solution), then we could verify that whether the certificate (the solution) is correct in polynomial time.

- Examples:
  - Circuit-SAT
  - Hamiltonian cycle
  - Graph coloring

- **NP** stands for "**N**ondeterministic **P**olynomial time".

2. P and NP

- $P \subseteq NP$

  since if a problem is in P, then we can solve it in polynomial time without even being given a certificate.

- Open problem:[1]

  Does $P \subset NP$ or $P = NP$?

[1] http://www.claymath.org/millennium-problems

2. P and NP

- The size of the input can change the classification of P or NP.

[2] CNF = Conjunctive Normal Form: a sequence of clauses separated by AND ($\wedge$) operator. A *clause* is a sequence of Boolean varilables separated by the Boolean OR ($\vee$) operator.

2. P and NP

- The size of the input can change the classification of P or NP.
- Examples:
  - Prime-testing problem:

$$O(n) \xrightarrow{n=10^m} O(10^m)$$

  - Knapsack problem

$$O(nW) \xrightarrow{W=10^m} O(n \cdot 10^m)$$

[2]CNF = Conjunctive Normal Form: a sequence of clauses separated by AND ($\wedge$) operator. A *clause* is a sequence of Boolean varilables separated by the Boolean OR ($\vee$) operator.

2. P and NP

- The size of the input can change the classification of P or NP.
- Examples:
  - Prime-testing problem:

$$O(n) \quad \overset{n=10^m}{\longrightarrow} \quad O(10^m)$$

  - Knapsack problem

$$O(nW) \quad \overset{W=10^m}{\longrightarrow} \quad O(n \cdot 10^m)$$

- Knowing the effect on complexity of the size of the input is important.

---

[2] CNF = Conjunctive Normal Form: a sequence of clauses separated by AND ($\wedge$) operator. A *clause* is a sequence of Boolean varilables separated by the Boolean OR ($\vee$) operator.

2. P and NP

- The size of the input can change the classification of P or NP.
- Examples:
  - Prime-testing problem:

$$O(n) \quad \stackrel{n=10^m}{\longrightarrow} \quad O(10^m)$$

  - Knapsack problem

$$O(nW) \quad \stackrel{W=10^m}{\longrightarrow} \quad O(n \cdot 10^m)$$

- Knowing the effect on complexity of the size of the input is important.
- Unfortunately, even with strong restrictions on the inputs, many NPC problems are still NPC.

  Example: 3-CNF SAT problem[2]

[2] CNF = Conjunctive Normal Form: a sequence of clauses separated by AND ($\wedge$) operator. A *clause* is a sequence of Boolean varilables separated by the Boolean OR ($\vee$) operator.

2. P and NP – recap

1. P and NP: formal definitions
2. Open problem: whether or not P is a proper subset of NP
3. The size of the input can change the classification of P or NP
   However, even with strong restrictions on the inputs, many NPC problems are still NPC.

3. NP-complete

- NP-complete (NPC) is the term used to describe decision problems that are the *hardest ones* in **NP** in the following sense

  > *If there were a polynomial-bounded algorithm for an NPC problem, then there would be a polynomial-bounded time for each problem in NP.*

3. NP-complete

Formal definition:

- A decision problem A is **NP-complete (NPC)** if

[3] Note: "NP-hard" does not mean "in NP and hard". It means "at least as hard as any problem in NP". Thus a problem can be NP-hard and not be in NP.

3. NP-complete

Formal definition:

- A decision problem A is **NP-complete (NPC)** if

  (1) $A \in \text{NP}$ and

[3] Note: "NP-hard" does not mean "in NP and hard". It means "at least as hard as any problem in NP". Thus a problem can be NP-hard and not be in NP.

3. NP-complete

Formal definition:

- A decision problem A is **NP-complete (NPC)** if

  (1) $A \in \text{NP}$ and

  (2) every other problems B in NP is *polynomially reducible* to A, denoted as

$$B \leq_T A$$

[3] Note: "NP-hard" does not mean "in NP and hard". It means "at least as hard as any problem in NP". Thus a problem can be NP-hard and not be in NP.

3. NP-complete

Formal definition:

- A decision problem A is **NP-complete (NPC)** if

  (1) $A \in$ NP and

  (2) every other problems B in NP is *polynomially reducible* to A, denoted as

$$B \leq_T A$$

If a problem satisfies the property (2), but not necessarily the property (1), we say the problem is NP-hard.[3]

[3] Note: "NP-hard" does not mean "in NP and hard". It means "at least as hard as any problem in NP". Thus a problem can be NP-hard and not be in NP.

3. NP-complete

Polynomial reduction

- Let A and B be two decision problems, B is polynomially reducible to A, if there is a poly-time computable transformation T such that

$$\text{Yes-instance of } A \quad \overset{\text{iff}}{\Longleftrightarrow} \quad \text{Yes-instance of } B$$

- Notation: $B \leq_T A$

3. NP-complete

- Cook's theorem (1971):[4]
  Circuit-SAT is NPC.
- Known NPC problems:
  - Graph coloring
  - Hamiltonian cycle
  - TSP
  - Knapsack
  - ... *see next page for more.*

[4] First result deomonstrating that a specific problem is NPC.

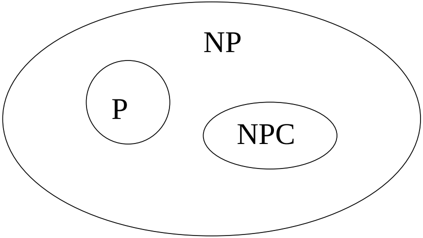
3. NP-complete

- Known NPC problems — *more*
  - Subset sum:
    Given a positive integer c, and a set $S = \{s_1, s_2, \ldots, s_n\}$ of positive integers s_i for $i = 1, 2, \ldots, n$. Assume that $\sum_{i=1}^{n} s_i \geq c$. Is there a subset $J \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in J} s_i = c$.
  - Bin packing problem:
    Suppose we have an unlimited number of bins, each of capacity 1, and n objects with sizes $s_1, s_2, \ldots, s_n$, where $0 < s_i \leq 1$. Determine the *smallest number* of bins into which objects can be packed.
  - Vertex cover problem:
    A **vertex-cover** of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$. The vertex-cover optimization problem is to find a vertex cover of minimum size.
  - Clique problem:
    A **clique** in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that each pair of V' is connected by an edge in E. The clique optimization problem is to find a clique of maximum size.

3. NP-complete

P, NP and NPC:

- How most theoretical computer scientists **view** the relationships among P, NP and NPC:
  - Both P and NPC are wholely contained within NP
  - P $\bigcap$ NPC $= \emptyset$



3. NP-complete – Recap

1. NP-complete (NPC): formal definition
2. Polynomial reduction
3. Cook's theorem
4. Examples of known NPC problems