lesson10 NP completeness

1.recall

1.**euler circuit problems**: find a path that touches every **edge** once

deep first search

2.**hamiton cycle**:find a single path that contains every **vertex**

hard!

3.**single-source unweighted shortest-path problem**

breadth fisrt search

4.**single-source unweighted longest-path problem**

hard! No known algorithms are guaranteed to run in polynomial time.

easiest: O(N) read inputs

hardest: undecidable problems

不是所有的true statement都可以被证明

2.eg:halting problem

Is it possible to have your C compiler detect all infinite loops?

No.

Proof: If there exists an infinite loop-checking program, then surely it could be used to check itself.

```
Loop( P )
{
/* 1 */ if ( P(P) loops )   print (YES);
/* 2 */ else infinite_loop();
}
```

P: program

if(loop(loop)

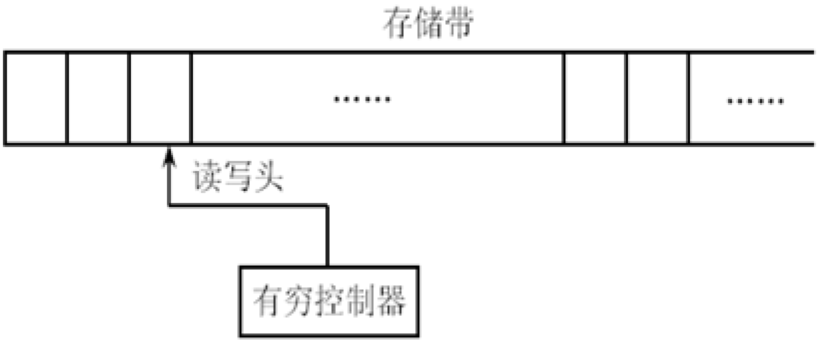
terminates,判断P是一个infinite_loop，

else loops 判断P是一个会terminate 矛盾!

3. the class NP

3.1turing machine

有穷控制器（有限状态机）、无穷带（符号集合）和读写头（读、改写、左移、右移）



图灵机是一个五元组（K，Σ，δ，s，H），其中：

- K 是有穷个状态的集合；
- Σ 是字母表，即符号的集合；
- s ∈K是初始状态；
- H∈K 是停机状态的集合，当控制器内部状态为停机状态时图灵机结束计算；
- δ是转移函数，即控制器的规则集合

operations:

改变state,改变head指向symbol的值，head每次移动一个位置

eg:

deterministic turing machine: 顺序执行instruction

non-deterministic turing machine：从有限集合中自由选择下一步，如果结果正确，就会一直选择这一步

3.2 NP

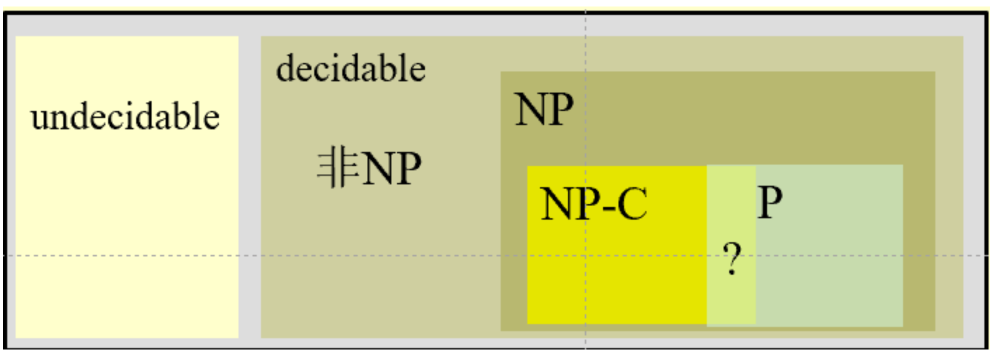
NP: Nondeterministic polynomial-time

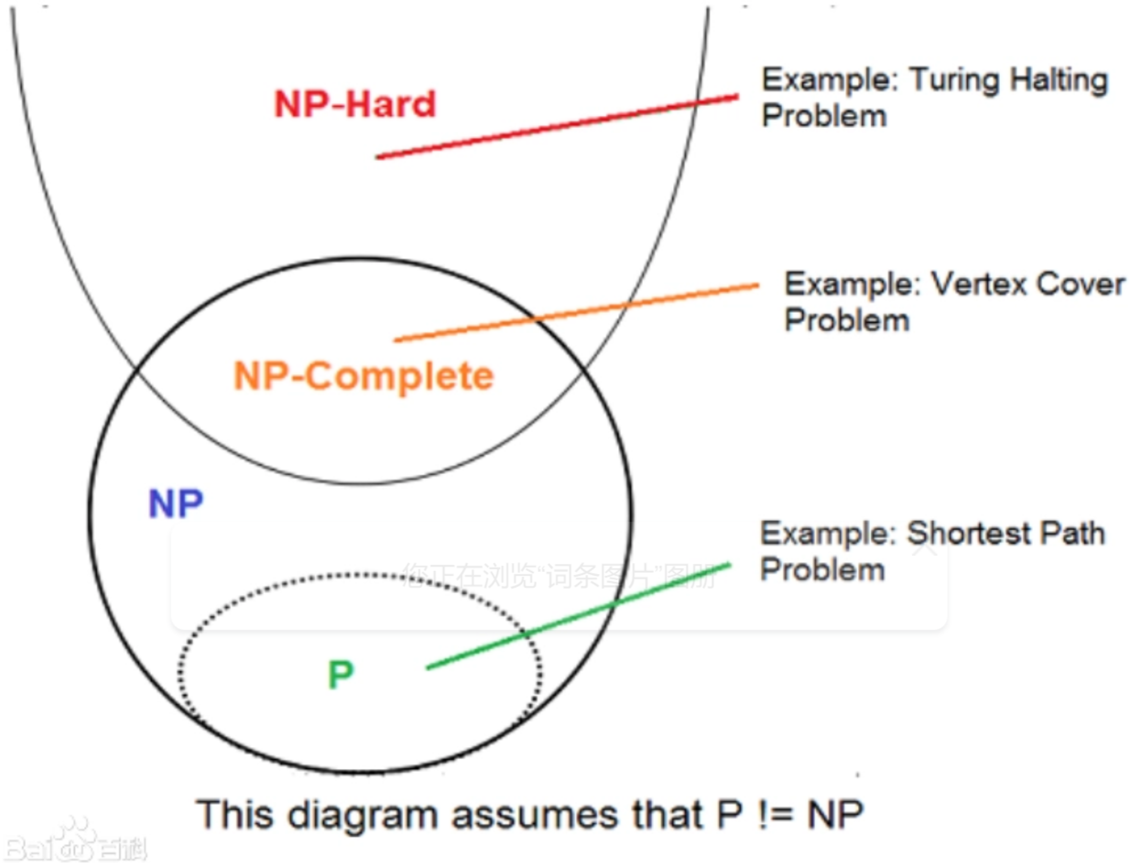
The problem is NP if we can prove any solution is true in polynomial time.

eg NP:Hamilton cycle problem

Note: Not all decidable problems are in NP. For example, consider the problem of determining whether a graph does not have a Hamiltonian cycle.

P∈NP，但是很多NP问题无法用P时间解决



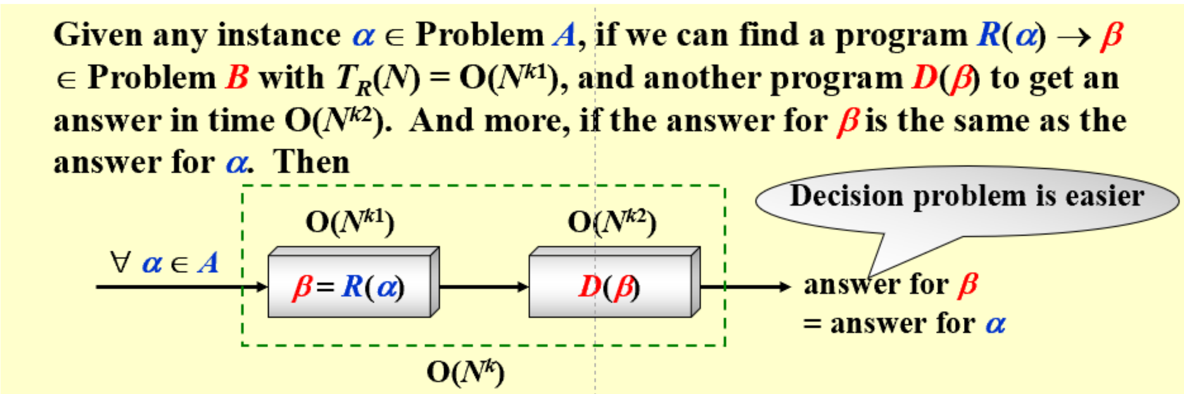


3.3 NP-completeness problem

An NP-complete problem has the property that any problem in NP can be polynomially reduced to it.

如果可以在polynomial time 解决NP问题，那么解决所有的NP问题，

也就是说要找到问题间的转换方式R



4.HCP to TSP

the **Hamiltonian cycle problem** is NP-complete. Prove that the **traveling salesman problem** is NP-complete as well.

proof:

Hamiltonian cycle problem:

Given a graph G=(V, E), is there a simple cycle that visits all vertices?

Traveling salesman problem:

Given a complete graph G=(V, E), with edge costs, and an integer K, is there a simple cycle that visits all vertices and has total cost ≤ K?

proof:

1)TSP属于NP

2)处理：每个已经存在的edge赋值1；连接所有的点，新的边赋值为2

3)G中存在一个hamilton cycle 等同于 G'中存在一个 salesman tour of total weight |V|

5.np-complete problem

the Satisfiability problem (Circuit-SAT): Input a boolean expression and ask if it has an assignment to the variables that gives the expression a value of 1.

6.formal-language framework

for decision problem

符号表示:

- An *alphabet* Σ is a finite set of symbols { 0, 1 }
- A *language* L over Σ is any set of strings made up of symbols from Σ $L = \{ x \in \Sigma^*: Q(x) = 1 \}$
- Denote *empty string* by ε
- Denote *empty language* by $\emptyset$
- Language of all strings over Σ is denoted by Σ*
- The *complement* of L is denoted by Σ^*-L
- The *concatenation* of two languages L_1 and L_2 is the language $L = \{ x_1x_2 : x_1 \in L_1 \text{ and } x_2 \in L_2 \}$.
- The *closure* or *Kleene star* of a language L is the language $L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup \cdots$, where L^k is the language obtained by concatenating L to itself k times

- Algorithm A *accepts* a string $x \in \{0, 1\}^*$ if $A(x) = 1$
- Algorithm A *rejects* a string x if $A(x) = 0$
- A language L is *decided* by an algorithm A if every binary string *in L* is *accepted* by A and every binary string *not in L* is *rejected* by A
- To *accept* a language, an algorithm need only worry about strings in L, but to *decide* a language, it must correctly accept or reject every string in $\{0, 1\}^*$

NP问题定义:

P = { $L \subseteq \{0, 1\}^*$: there exists an algorithm A that decides L in polynomial time }

7.verification algorithm

A two-argument algorithm A verifies an input string x if there exists a certificate y such that A(x, y) = 1.

〖**Example**〗 **For SAT**

$$x = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4)$$

Certificate: $y = \{ x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1 \}$