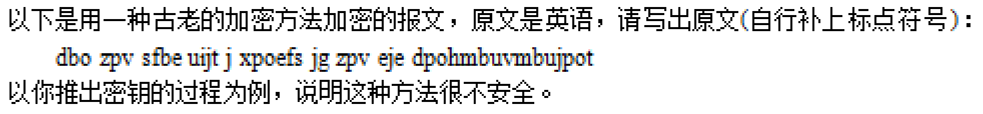
- 期末试题讲解

第1题

- dbo zpv sfbe uijt j xpoefs jg zpv eje dpohmbuvmbujpot
  - j → a, i
- Can you read this, I wonder? If you did, congratulations!



以下是用一种古老的加密方法加密的报文，原文是英语，请写出原文(自行补上标点符号)：

dbo zpv sfbe uijt j xpoefs jg zpv eje dpohmbuvmbujpot

以你推出密钥的过程为例，说明这种方法很不安全。

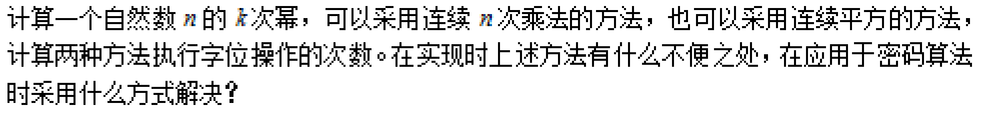
第2题

- 计算一个自然数n的k次幂，可以采用连续(k-1)次乘法的方法，也可以采用连续平方的方法，计算两种方法执行字位操作的次数。在实现时上述方法有什么不便之处，在应用于密码算法时采用什么方式解决？
  - 假设n的位数为β，连续相乘位操作

$$\beta^2+2\,\beta^2+3\,\beta^2+\ldots+(k-1)\,\beta^2$$

  - 连续平方法执行乘法运算的次数为$\lfloor\log_2 k\rfloor + k - 2^{\lfloor\log_2 k\rfloor}$，因此，位操作次数为

$$O\left(\left(\lfloor\log_2 k\rfloor + k - 2^{\lfloor\log_2 k\rfloor}\right)\beta^2\right)$$


计算一个自然数 n 的 k 次幂，可以采用连续 n 次乘法的方法，也可以采用连续平方的方法，计算两种方法执行字位操作的次数。在实现时上述方法有什么不便之处，在应用于密码算法时采用什么方式解决？

第3题

- 设计一个算法计算两个复数的乘积：(a+b*i*)(c+d*i*), 只用3次乘法。
  - 多种解法
- 例如
  - A=(a+b)(c+d)
  - B=ac
  - C=bd
  - (a+b*i*)(c+d*i*)=ac-bd+(ad+bc)*i*=B-C+(A-B-C)*i*

设计一个算法计算两个复数的乘积：(a+b*i*)(c+d*i*)，只用 3 次乘法。

第4题

- 给出一个线性算法，在树中找到最大匹配。说明你的结果为什么一定是最大匹配。

- 在树中DP求解。**状态转移方程：**

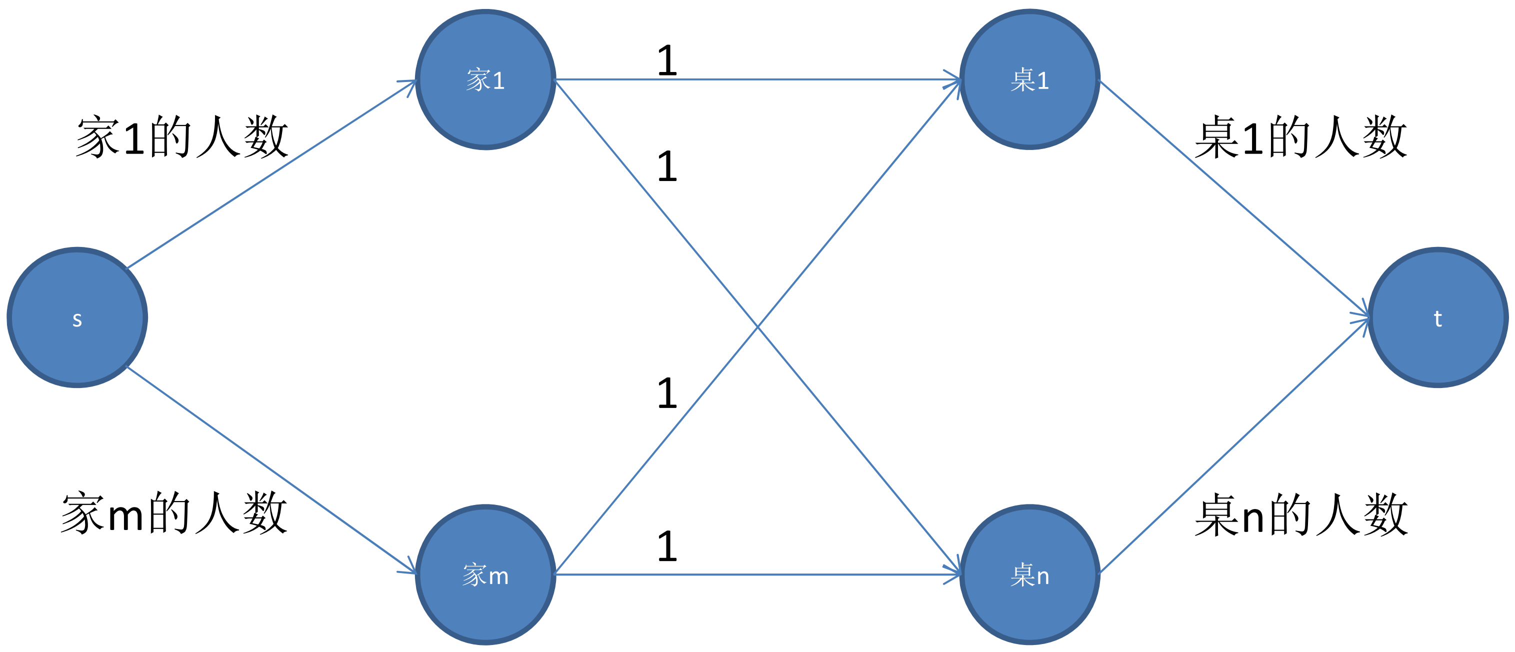
$$F[i][0] = \sum_{v_j \in R(i)} \left(\max\{F[j][0], F[j][1]\} \right)$$

$$F[i][1] = \max_{v_j \in R(i)} \left\{ F[j][0] + \sum_{k \in R(i) \wedge k \neq j} \left(\max\{F[k][0], F[k][1]\} \right) \right\} + 1$$

- 对于树的根节点v_{r}，树的最大匹配为$\max\{F[r][0], F[r][1]\}$。

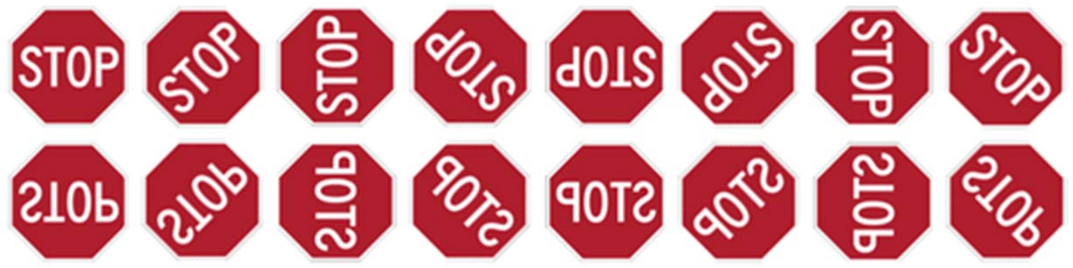
给出一个线性算法，在树中找到最大匹配。说明你的结果为什么一定是最大匹配。

第5题



m 个家庭一起外出野餐，有 n 个桌子。为了加强交流，希望同一家庭成员一定不坐在一张桌子上。每个家庭人数为 a_i $(i=1,2,\ldots m)$; 每张桌子可以坐 b_j $(j=1,2, \ldots, n)$个人。建立一个网络流模型解决如何安排每个人坐的桌次的问题。
这个问题是否一定有可行解，如果不是，你的方法能否判断无解？

第6题



旋转

对称

±60度旋转：6点的标号必须一样，则各有3种标号在旋转之后不变

±120度旋转：2组3点的标号必须一样，则各有3^2种标号在旋转之后不变

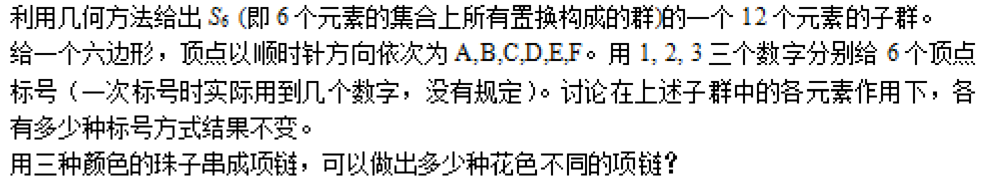
180度旋转：3组2点的标号必须一样，则各有3^3种标号在旋转之后不变

0度旋转：任意3^6种标号在旋转之后不变

对角线对称：2组2点的标号必须一样，则各有3^4种标号在对称后不变

边中点连线对称：3组2点的标号必须一样，则各有3^3种标号在对称后不变

92种不同的项链：Polya计数定理



利用几何方法给出 S_6 (即 6 个元素的集合上所有置换构成的群)的一个 12 个元素的子群。
给一个六边形，顶点以顺时针方向依次为 A,B,C,D,E,F。用 1, 2, 3 三个数字分别给 6 个顶点标号（一次标号时实际用到几个数字，没有规定）。讨论在上述子群中的各元素作用下，各有多少种标号方式结果不变。
用三种颜色的珠子串成项链，可以做出多少种花色不同的项链？

第7题

1. Arrange the vertices arbitrarily into a cycle, ignoring adjacencies in the graph.
2. While the cycle contains two consecutive vertices v_i and v_{i+1} that are not adjacent in the graph, perform the following two steps:
   - Search for an index j such that the four vertices v_i, v_{i+1}, v_j, and v_{j+1} are all distinct and such that the graph contains edges from v_i to v_j and from v_{j+1} to v_{i+1}
   - Reverse the part of the cycle between v_{i+1} and v_j (inclusive).

Each step increases the number of consecutive pairs in the cycle that are adjacent in the graph, by one or two pairs (depending on whether v_j and v_{j+1} are already adjacent), so the outer loop can only happen at most n times before the algorithm terminates, where n is the number of vertices in the given graph. By an argument similar to the one in the proof of the theorem, the desired index j must exist, or else the nonadjacent vertices v_i and v_{i+1} would have too small a total degree. Finding i and j, and reversing part of the cycle, can all be accomplished in time $O(n)$. Therefore, the total time for the algorithm is $O(n^2)$, matching the number of edges in the input graph.

我们知道：任意一个至少3个顶点的简单图，若任意两个不相邻的顶点的度数之和不小于图中顶点的总数，则该图是哈密尔顿图。你能否设计一个算法对于满足这一条件的图找出一条哈密尔顿回路？

- 教材讨论
  - JH第2章第3节

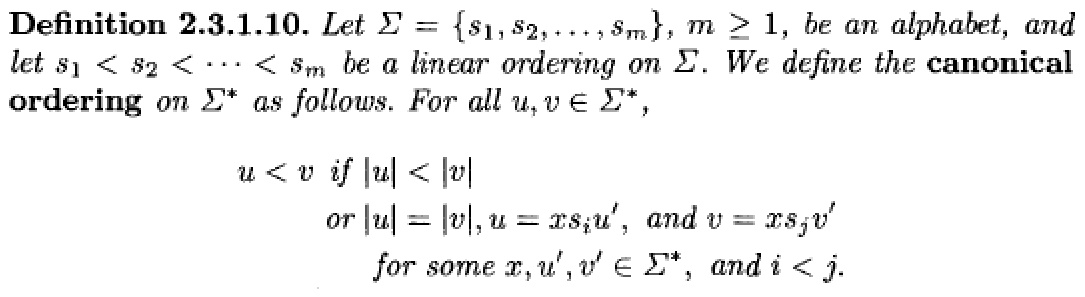
问题1：字母表、词、语言

- alphabet、symbol、word、language
  - 它们是如何被形式化定义的？
  - 你能举出一些实际生活中的例子吗？

- 你能设计一种语言来编码全班同学问求的期末考试成绩吗？
  - 你设计的字母表、词和语言分别是什么？
- 你能设计一种语言来编码图片吗？
  - 你设计的字母表、词和语言分别是什么？
- 编码视频呢？
  - 你设计的字母表、词和语言分别是什么？

- concatenation of word/language、subword/prefix/suffix
  - 你自己能够给出它们的形式化定义吗？
  - （然后再看看与书上的有什么不同）

问题1：字母表、词、语言 (续)

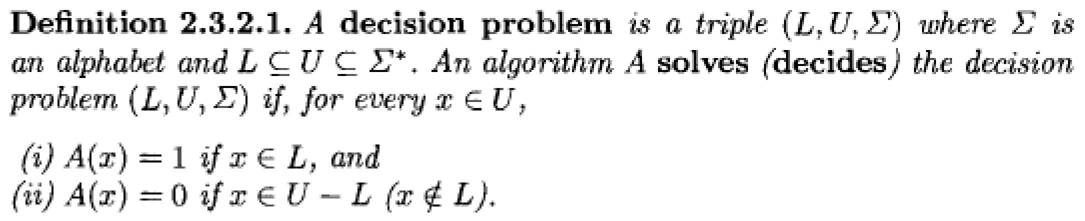


Definition 2.3.1.10. *Let* $\Sigma = \{s_1, s_2, \ldots, s_m\}$, $m \geq 1$, *be an alphabet, and let* $s_1 < s_2 < \cdots < s_m$ *be a linear ordering on* Σ. *We define the* **canonical ordering** *on* Σ^* *as follows. For all* $u, v \in \Sigma^*$,

$$\begin{aligned} u < v \ \textit{if}\ & |u| < |v| \\ \textit{or}\ & |u| = |v|, u = xs_iu', \textit{ and } v = xs_jv' \\ & \textit{for some } x, u', v' \in \Sigma^*, \textit{ and } i < j. \end{aligned}$$

- 我们为什么需要一种词的排序规则？
- 你能解释这条排序规则吗？
- 你能利用这条规则，给出一个字符串排序算法吗？
  - 仔细想想，这条规则被你用了几次？

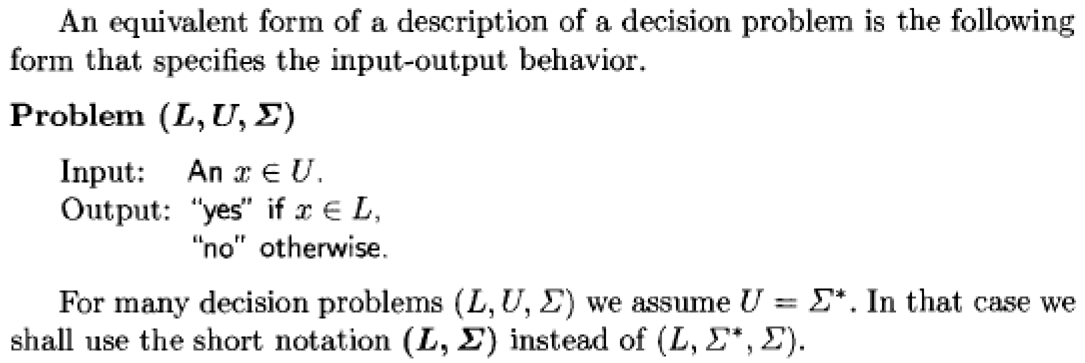
问题2：判定和优化问题



Definition 2.3.2.1. *A* **decision problem** *is a triple* (L, U, Σ) *where* Σ *is an alphabet and* $L \subseteq U \subseteq \Sigma^*$. *An algorithm* A **solves (decides)** *the decision problem* (L, U, Σ) *if, for every* $x \in U$,

(i) $A(x) = 1$ *if* $x \in L$, *and*
(ii) $A(x) = 0$ *if* $x \in U - L$ $(x \notin L)$.

- decision problem中的三个符号分别表示什么意思？
  - 这里的word是什么？
- 判定算法应该给出怎样的结果？



An equivalent form of a description of a decision problem is the following form that specifies the input-output behavior.

Problem $(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{\Sigma})$

Input: An $x \in U$.
Output: "yes" if $x \in L$,
"no" otherwise.

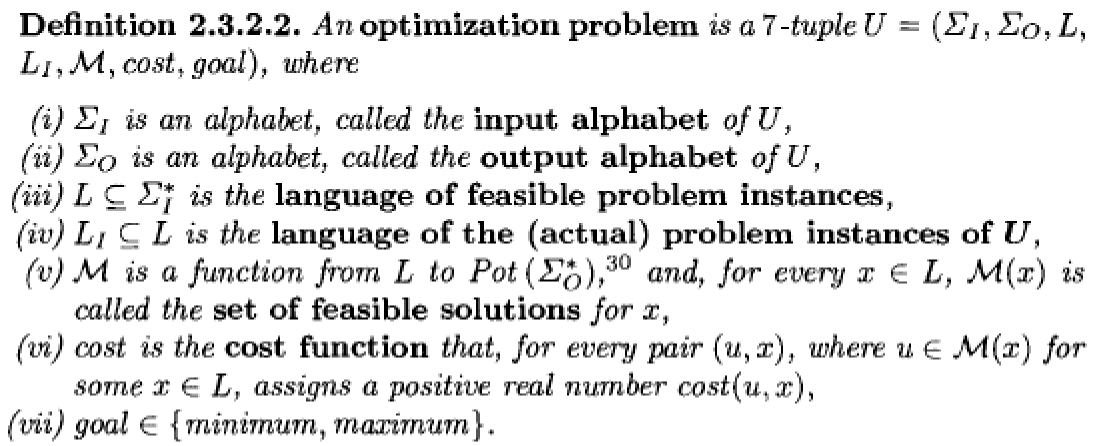
For many decision problems (L, U, Σ) we assume $U = \Sigma^*$. In that case we shall use the short notation $(\boldsymbol{L}, \boldsymbol{\Sigma})$ instead of (L, Σ^*, Σ).

- 你理解这段话的含义了吗？

问题2：判定和优化问题 (续)

- 这些问题分别是什么含义？它们的L分别是什么？你能理解或给出形式化定义吗？
  - Primality testing $\{w \in \{0,1\}^* \mid Number(w) \text{ is a prime}\}$
  - Equivalence problem for polynomials
  - Satisfiability problem $\{w \in \Sigma_{logic}^{+} \mid w \text{ is a code of a satisfiable formula in CNF}\}$
  - Clique problem $\{x\#w \in \{0,1,\#\}^* \mid x \in \{0,1\}^* \text{ and } w \text{ represents a graph that contains a clique of size } Number(x)\}$
  - Vertex cover problem $\{u\#w \in \{0,1,\#\}^+ \mid u \in \{0,1\}^+ \text{ and } w \text{ represents a graph that contains a vertex cover of size } Number(u)\}$
  - Hamiltonian cycle problem $\{w \in \{0,1,\#\}^* \mid w \text{ represents a graph that contains a Hamiltonian cycle}\}$
  - Existence of a solution of linear integer programming
    $\{\langle A,b\rangle \in \{0,1,\#\}^* \mid Sol_{\mathbb{Z}}(A,b) \neq \emptyset\}$

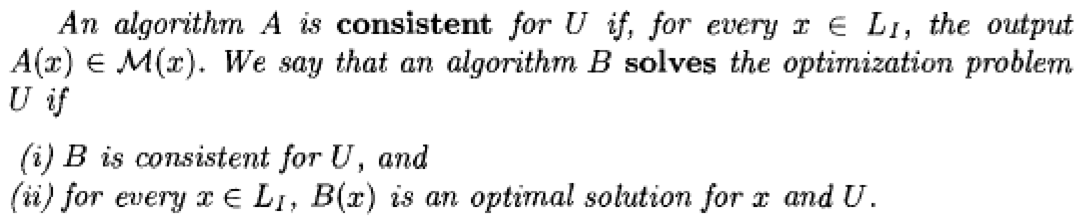
问题2：判定和优化问题 (续)



Definition 2.3.2.2. *An* **optimization problem** *is a 7-tuple* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$, *where*

(i) Σ_I *is an alphabet, called the* **input alphabet** *of* U,
(ii) Σ_O *is an alphabet, called the* **output alphabet** *of* U,
(iii) $L \subseteq \Sigma_I^*$ *is the* **language of feasible problem instances**,
(iv) $L_I \subseteq L$ *is the* **language of the (actual) problem instances of** U,
(v) $\mathcal{M}$ *is a function from* L *to* $Pot(\Sigma_O^*)$,[30] *and, for every* $x \in L$, $\mathcal{M}(x)$ *is called the* **set of feasible solutions** *for* x,
(vi) $cost$ *is the* **cost function** *that, for every pair* (u, x), *where* $u \in \mathcal{M}(x)$ *for some* $x \in L$, *assigns a positive real number* $cost(u, x)$,
(vii) $goal \in \{minimum, maximum\}$.

- optimization problem中的七个符号分别表示什么意思？



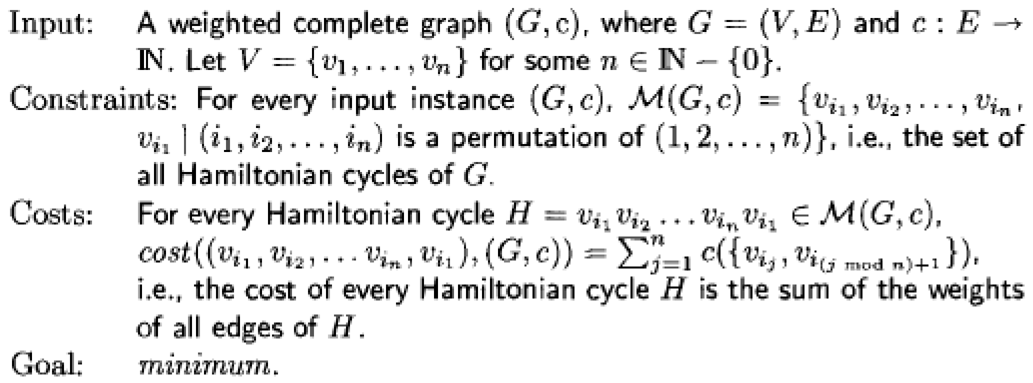
An algorithm A *is* **consistent** *for* U *if, for every* $x \in L_I$, *the output* $A(x) \in \mathcal{M}(x)$. *We say that an algorithm* B **solves** *the optimization problem* U *if*

(i) B *is consistent for* U, *and*
(ii) for every $x \in L_I$, $B(x)$ *is an optimal solution for* x *and* U.

- 优化算法应该给出怎样的结果？

问题2：判定和优化问题 (续)

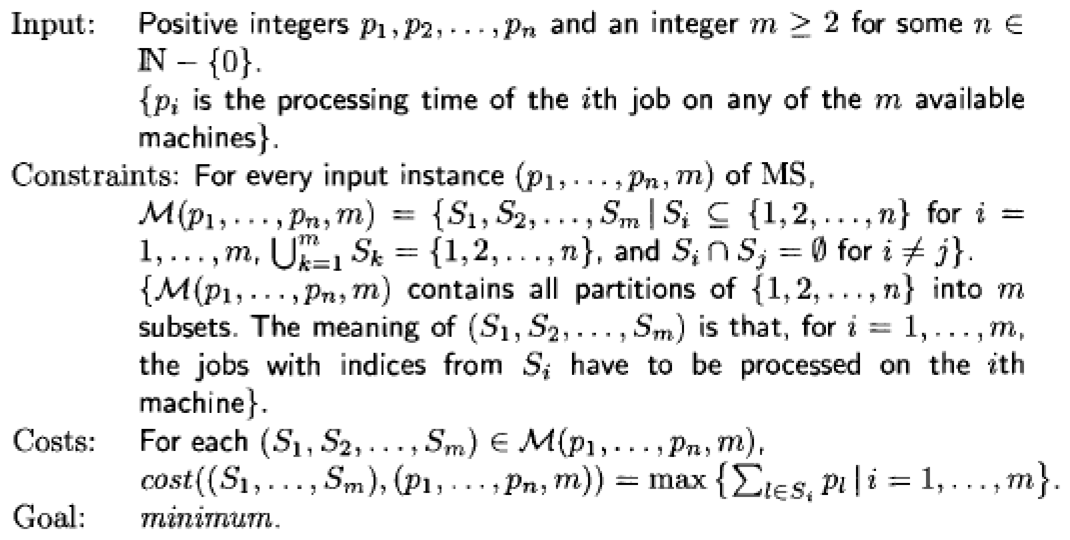
- 你理解traveling salesperson problem了吗？



Input:	A weighted complete graph (G, c), where $G = (V, E)$ and $c : E \rightarrow \mathbb{N}$. Let $V = \{v_1, \dots, v_n\}$ for some $n \in \mathbb{N} - \{0\}$.
Constraints:	For every input instance (G, c), $\mathcal{M}(G, c) = \{v_{i_1}, v_{i_2}, \dots, v_{i_n}, v_{i_1} \mid (i_1, i_2, \dots, i_n)$ is a permutation of $(1, 2, \dots, n)\}$, i.e., the set of all Hamiltonian cycles of G.
Costs:	For every Hamiltonian cycle $H = v_{i_1} v_{i_2} \dots v_{i_n} v_{i_1} \in \mathcal{M}(G, c)$, $cost((v_{i_1}, v_{i_2}, \dots v_{i_n}, v_{i_1}), (G, c)) = \sum_{j=1}^{n} c(\{v_{i_j}, v_{i_{(j \bmod n)+1}}\})$, i.e., the cost of every Hamiltonian cycle H is the sum of the weights of all edges of H.
Goal:	*minimum*.

问题2：判定和优化问题 (续)

- 你理解makespan scheduling problem了吗？



Input: Positive integers $p_1, p_2, \ldots, p_n$ and an integer $m \geq 2$ for some $n \in \mathbb{N} - \{0\}$.
$\{p_i$ is the processing time of the ith job on any of the m available machines$\}$.

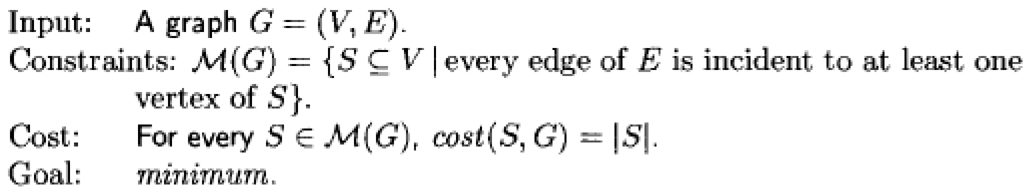
Constraints: For every input instance $(p_1, \ldots, p_n, m)$ of MS,
$\mathcal{M}(p_1, \ldots, p_n, m) = \{S_1, S_2, \ldots, S_m \mid S_i \subseteq \{1, 2, \ldots, n\}$ for $i = 1, \ldots, m$, $\bigcup_{k=1}^{m} S_k = \{1, 2, \ldots, n\}$, and $S_i \cap S_j = \emptyset$ for $i \neq j\}$.
$\{\mathcal{M}(p_1, \ldots, p_n, m)$ contains all partitions of $\{1, 2, \ldots, n\}$ into m subsets. The meaning of $(S_1, S_2, \ldots, S_m)$ is that, for $i = 1, \ldots, m$, the jobs with indices from S_i have to be processed on the ith machine$\}$.

Costs: For each $(S_1, S_2, \ldots, S_m) \in \mathcal{M}(p_1, \ldots, p_n, m)$,
$cost((S_1, \ldots, S_m), (p_1, \ldots, p_n, m)) = \max \{\sum_{l \in S_i} p_l \mid i = 1, \ldots, m\}$.

Goal: *minimum*.

问题2：判定和优化问题 (续)

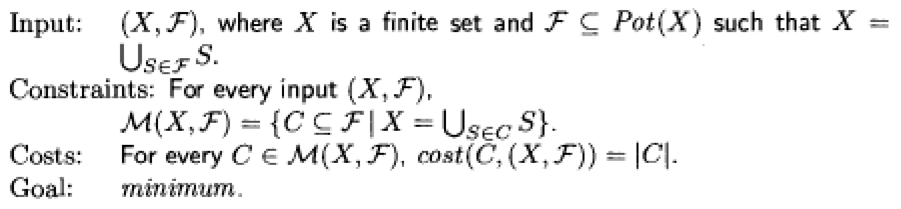
- 你理解minimum vertex cover problem了吗？



Input:	A graph $G = (V, E)$.
Constraints:	$\mathcal{M}(G) = \{S \subseteq V \mid$ every edge of E is incident to at least one vertex of $S\}$.
Cost:	For every $S \in \mathcal{M}(G)$, $cost(S, G) = \lvert S \rvert$.
Goal:	*minimum*.

问题2：判定和优化问题 (续)

- 你理解set cover problem了吗？



Input: $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq Pot(X)$ such that $X = \bigcup_{S \in \mathcal{F}} S$.

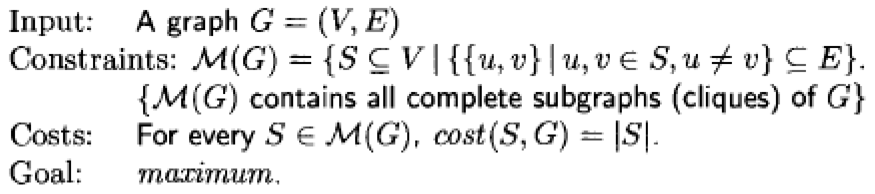
Constraints: For every input $(X, \mathcal{F})$,
$\mathcal{M}(X, \mathcal{F}) = \{C \subseteq \mathcal{F} \mid X = \bigcup_{S \in C} S\}$.

Costs: For every $C \in \mathcal{M}(X, \mathcal{F})$, $cost(C, (X, \mathcal{F})) = |C|$.

Goal: $minimum$.

问题2：判定和优化问题 (续)

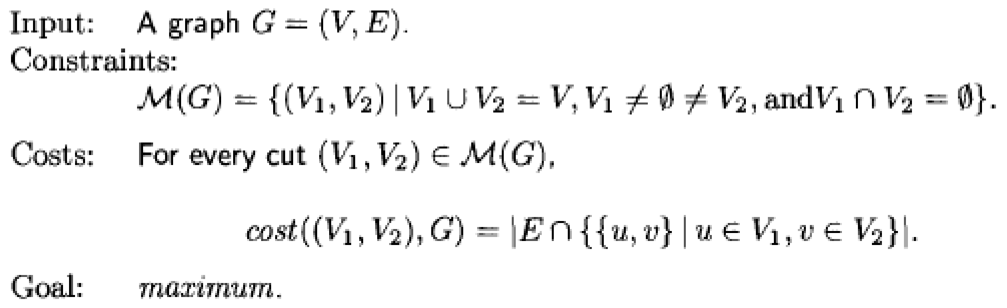
- 你理解maximum clique problem了吗？



Input:	A graph $G = (V, E)$
Constraints:	$\mathcal{M}(G) = \{S \subseteq V \mid \{\{u, v\} \mid u, v \in S, u \neq v\} \subseteq E\}$.
	$\{\mathcal{M}(G)$ contains all complete subgraphs (cliques) of $G\}$
Costs:	For every $S \in \mathcal{M}(G)$, $cost(S, G) = \lvert S \rvert$.
Goal:	*maximum*.

问题2：判定和优化问题 (续)

- 你理解maximum cut problem了吗？



Input: A graph $G = (V, E)$.

Constraints:

$\mathcal{M}(G) = \{(V_1, V_2) \mid V_1 \cup V_2 = V, V_1 \neq \emptyset \neq V_2, \text{and} V_1 \cap V_2 = \emptyset\}$.

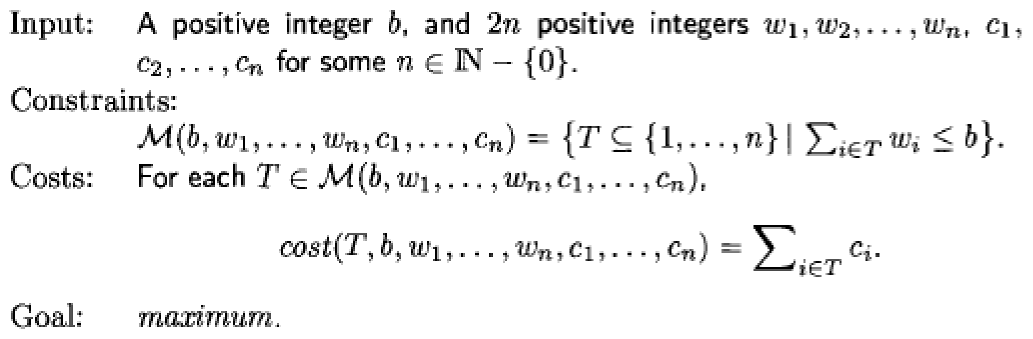
Costs: For every cut $(V_1, V_2) \in \mathcal{M}(G)$,

$$cost((V_1, V_2), G) = |E \cap \{\{u, v\} \mid u \in V_1, v \in V_2\}|.$$

Goal: *maximum*.

问题2：判定和优化问题 (续)

- 你理解knapsack problem了吗？



Input: A positive integer b, and $2n$ positive integers $w_1, w_2, \ldots, w_n$, c_1, $c_2, \ldots, c_n$ for some $n \in \mathbb{N} - \{0\}$.

Constraints:
$\mathcal{M}(b, w_1, \ldots, w_n, c_1, \ldots, c_n) = \{T \subseteq \{1, \ldots, n\} \mid \sum_{i \in T} w_i \leq b\}$.

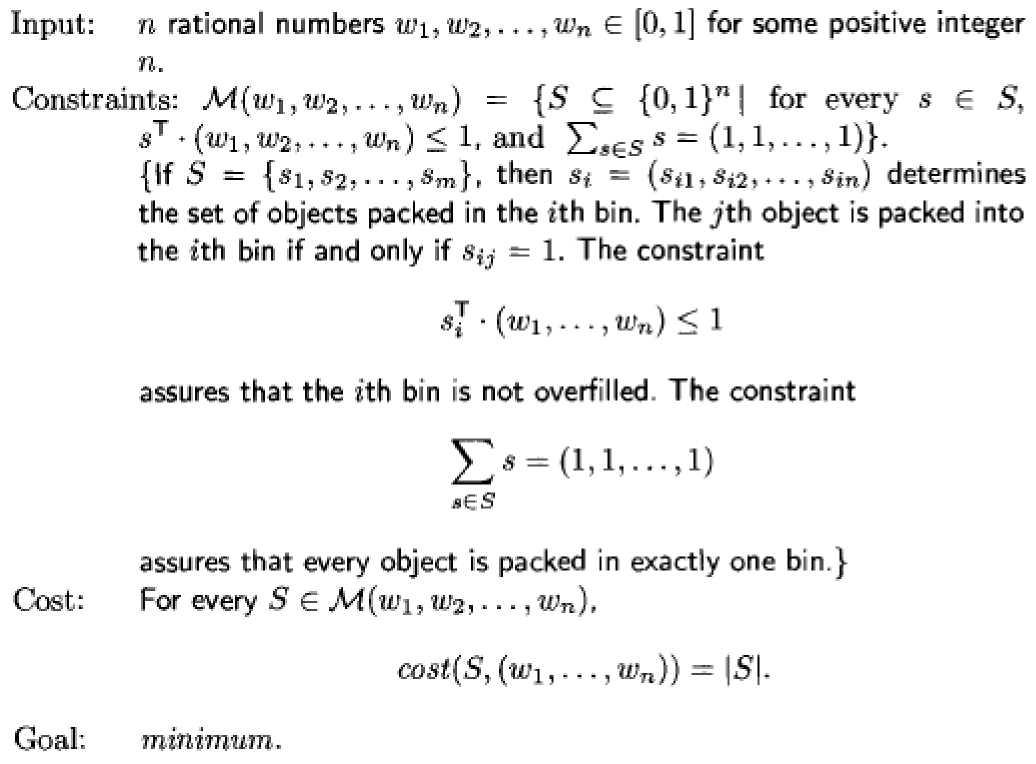
Costs: For each $T \in \mathcal{M}(b, w_1, \ldots, w_n, c_1, \ldots, c_n)$,

$$cost(T, b, w_1, \ldots, w_n, c_1, \ldots, c_n) = \sum_{i \in T} c_i.$$

Goal: *maximum*.

问题2：判定和优化问题 (续)

- 你理解bin-packing problem了吗？



Input: n rational numbers $w_1, w_2, \ldots, w_n \in [0,1]$ for some positive integer n.

Constraints: $\mathcal{M}(w_1, w_2, \ldots, w_n) = \{S \subseteq \{0,1\}^n \mid$ for every $s \in S$, $s^{\mathsf{T}} \cdot (w_1, w_2, \ldots, w_n) \leq 1$, and $\sum_{s \in S} s = (1,1,\ldots,1)\}$.
{If $S = \{s_1, s_2, \ldots, s_m\}$, then $s_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ determines the set of objects packed in the ith bin. The jth object is packed into the ith bin if and only if $s_{ij} = 1$. The constraint

$$s_i^{\mathsf{T}} \cdot (w_1, \ldots, w_n) \leq 1$$

assures that the ith bin is not overfilled. The constraint

$$\sum_{s \in S} s = (1,1,\ldots,1)$$

assures that every object is packed in exactly one bin.}

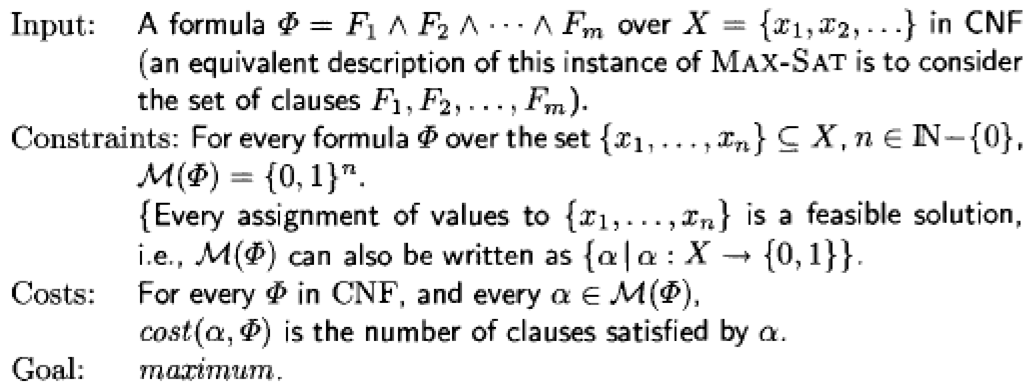
Cost: For every $S \in \mathcal{M}(w_1, w_2, \ldots, w_n)$,

$$cost(S, (w_1, \ldots, w_n)) = |S|.$$

Goal: *minimum*.

问题2：判定和优化问题 (续)

- 你理解maximum satisfiability problem了吗？



Input: A formula $\Phi = F_1 \wedge F_2 \wedge \cdots \wedge F_m$ over $X = \{x_1, x_2, \ldots\}$ in CNF (an equivalent description of this instance of MAX-SAT is to consider the set of clauses $F_1, F_2, \ldots, F_m$).

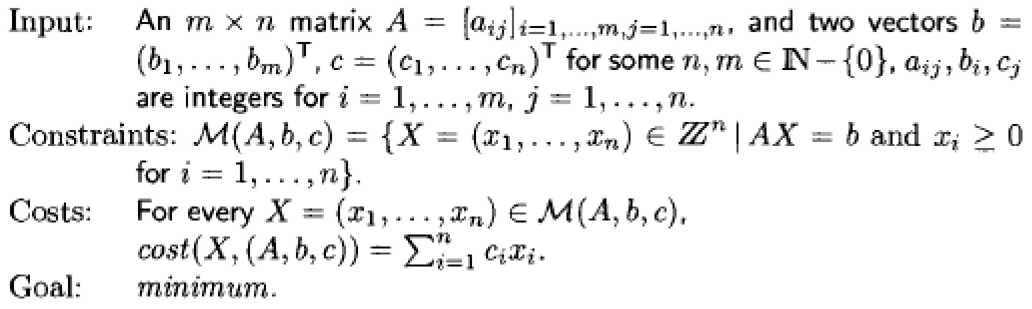
Constraints: For every formula Φ over the set $\{x_1, \ldots, x_n\} \subseteq X$, $n \in \mathbb{N} - \{0\}$, $\mathcal{M}(\Phi) = \{0,1\}^n$.
{Every assignment of values to $\{x_1, \ldots, x_n\}$ is a feasible solution, i.e., $\mathcal{M}(\Phi)$ can also be written as $\{\alpha \mid \alpha : X \to \{0,1\}\}$}.

Costs: For every Φ in CNF, and every $\alpha \in \mathcal{M}(\Phi)$, $cost(\alpha, \Phi)$ is the number of clauses satisfied by α.

Goal: *maximum*.

问题2：判定和优化问题 (续)

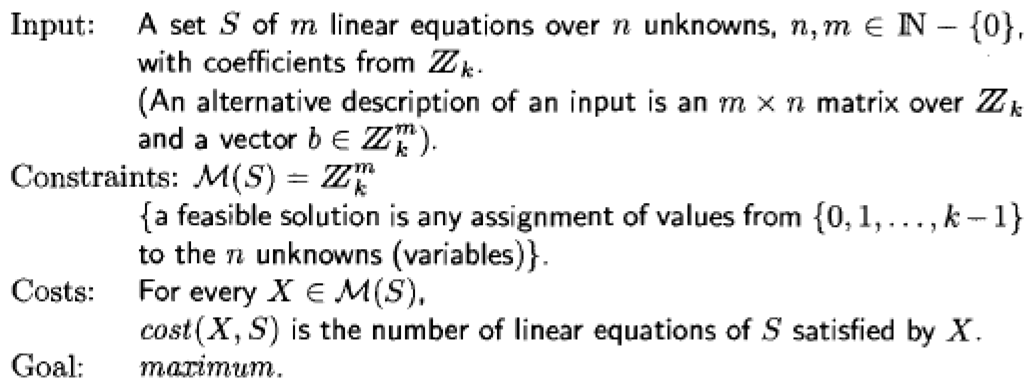
- 你理解integer linear programming了吗？



Input:	An $m \times n$ matrix $A = [a_{ij}]_{i=1,\dots,m,j=1,\dots,n}$, and two vectors $b = (b_1, \dots, b_m)^\mathsf{T}$, $c = (c_1, \dots, c_n)^\mathsf{T}$ for some $n, m \in \mathbb{N} - \{0\}$, a_{ij}, b_i, c_j are integers for $i = 1, \dots, m$, $j = 1, \dots, n$.
Constraints:	$\mathcal{M}(A, b, c) = \{X = (x_1, \dots, x_n) \in \mathbb{Z}^n \mid AX = b$ and $x_i \geq 0$ for $i = 1, \dots, n\}$.
Costs:	For every $X = (x_1, \dots, x_n) \in \mathcal{M}(A, b, c)$, $cost(X, (A, b, c)) = \sum_{i=1}^{n} c_i x_i$.
Goal:	*minimum*.

问题2：判定和优化问题 (续)

- 你理解maximum linear equation problem mod *k*了吗？



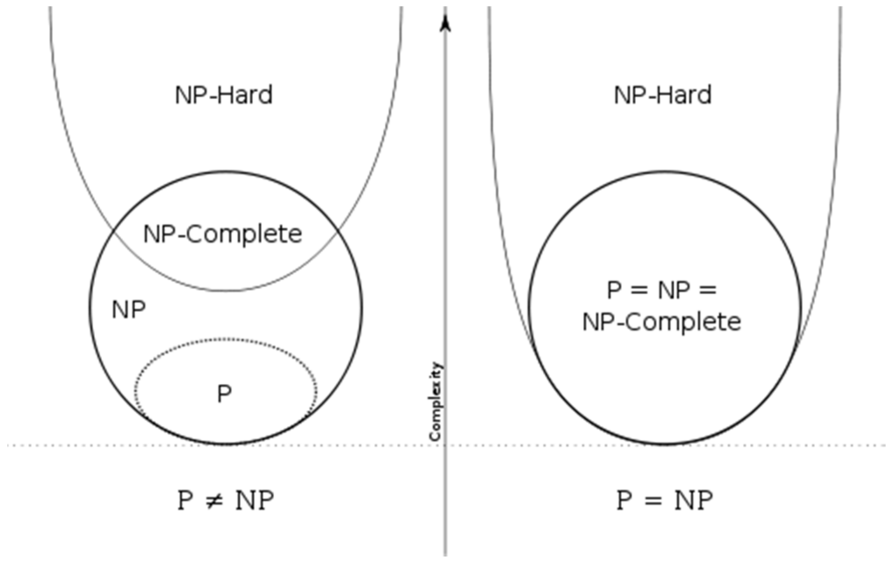
Input:	A set S of m linear equations over n unknowns, $n, m \in \mathbb{N} - \{0\}$, with coefficients from $\mathbb{Z}_k$. (An alternative description of an input is an $m \times n$ matrix over $\mathbb{Z}_k$ and a vector $b \in \mathbb{Z}_k^m$).
Constraints:	$\mathcal{M}(S) = \mathbb{Z}_k^m$ $\{$a feasible solution is any assignment of values from $\{0, 1, \ldots, k-1\}$ to the n unknowns (variables)$\}$.
Costs:	For every $X \in \mathcal{M}(S)$, $cost(X, S)$ is the number of linear equations of S satisfied by X.
Goal:	*maximum*.

问题3：P和NP

- 你怎么理解upper bound和lower bound？

问题3：P和NP (续)

- 你能解释清楚这些概念及其之间的关系吗？
  - P
  - NP
  - NP-hard
  - NP-complete



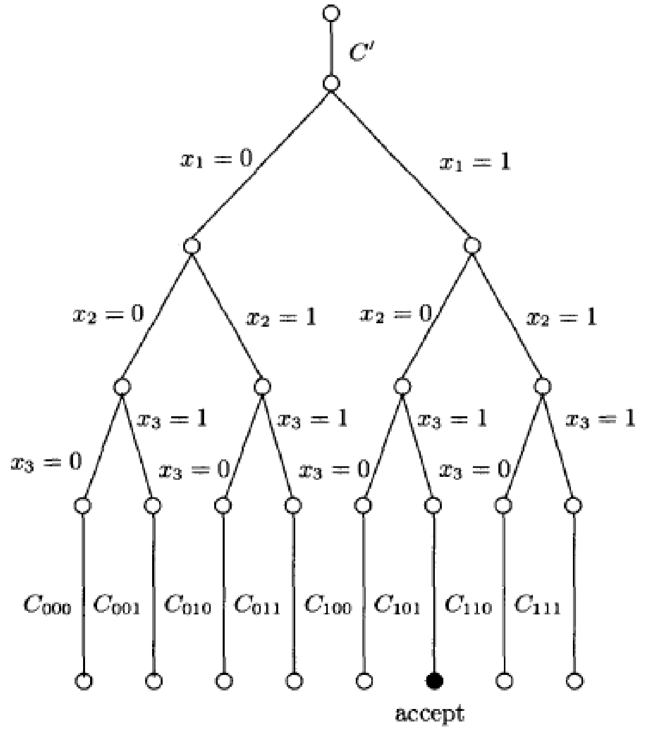
- 注意，经常被忽视的一点是，严格来说
  - P、NP、NP-complete都是描述判定问题的
  - NP-hard可以描述判定问题、优化问题等各类问题

问题3：P和NP (续)

- 你怎么理解这段话？

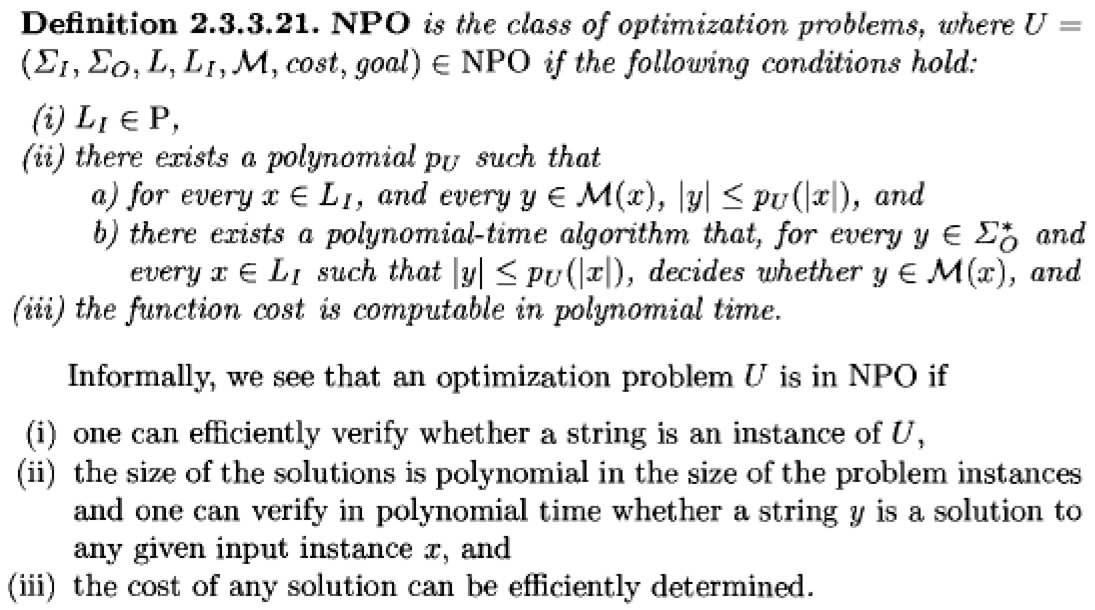
The complexity of deterministic computations is the complexity of proving the correctness of the produced output, while the complexity of nondeterministic computation is equivalent to the complexity of deterministic verification of a given proof (certificate) of the fact $x \in L$.

- 你能结合这个例子来解释吗？



问题3：P和NP (续)

- 优化问题也有自己的“NP”和“P”，你能理解吗？

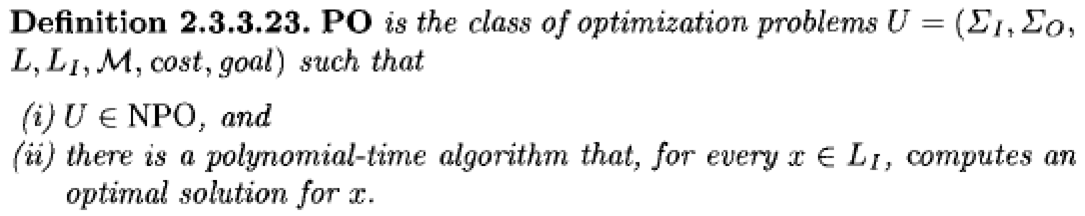


Definition 2.3.3.21. NPO *is the class of optimization problems, where* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal) \in \mathrm{NPO}$ *if the following conditions hold:*

(i) $L_I \in \mathrm{P}$,
(ii) there exists a polynomial p_U *such that*
  a) for every $x \in L_I$, *and every* $y \in \mathcal{M}(x)$, $|y| \leq p_U(|x|)$, *and*
  b) there exists a polynomial-time algorithm that, for every $y \in \Sigma_O^*$ *and every* $x \in L_I$ *such that* $|y| \leq p_U(|x|)$, *decides whether* $y \in \mathcal{M}(x)$, *and*
(iii) the function cost is computable in polynomial time.

Informally, we see that an optimization problem U is in NPO if

(i) one can efficiently verify whether a string is an instance of U,
(ii) the size of the solutions is polynomial in the size of the problem instances and one can verify in polynomial time whether a string y is a solution to any given input instance x, and
(iii) the cost of any solution can be efficiently determined.



Definition 2.3.3.23. PO *is the class of optimization problems* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$ *such that*

(i) $U \in \mathrm{NPO}$, *and*
(ii) there is a polynomial-time algorithm that, for every $x \in L_I$, *computes an optimal solution for* x.