I216 計算量の理論と離散数学

上原隆平、面　和成

I216 Computational Complexity and Discrete Mathematics

by
Prof. Ryuhei Uehara
and
Prof. Kazumasa Omote

計算量の理論

- ゴール1:
  - *“計算可能な関数/問題/言語/集合”*

- ゴール2:
  - 「問題の困難さ」を示す方法を学ぶ
    - 計算可能な問題であっても、*手におえない*場合がある！
      - 計算に必要な資源（時間・領域）が多すぎるとき
    - 関連する専門用語;

      クラスNP, P≠NP予想, NP困難性, 還元

Computational Complexity

- Goal 1:
  - *"Computable Function/Problem/Language/Set"*

- Goal 2:
  - How can you show "*Difficulty of Problem*"
    - There are *intractable* problems even if they are computable!
      - because they require too many resources (time/space)!
    - Technical terms;

      The class NP, P≠NP conjecture, NP-hardness, reduction

5.計算量の理論

5.1.計算時間の評価

5.1.3.問題の時間計算量

定義: 自然数上の関数 $t(n)$ に対して，時間計算量 $O(t(n))$ の集合(認識問題)全体の集合を **$O(t(n))$時間計算量クラス** とよび，$\mathrm{TIME}(t(n))$ とかく．こうした関数 $t(n)$ は制限時間と呼ぶ．

例1 PRIME は $\mathrm{TIME}(n^2 2^n)$ の要素であったが
今は $\mathrm{TIME}(n^6)$ の要素．

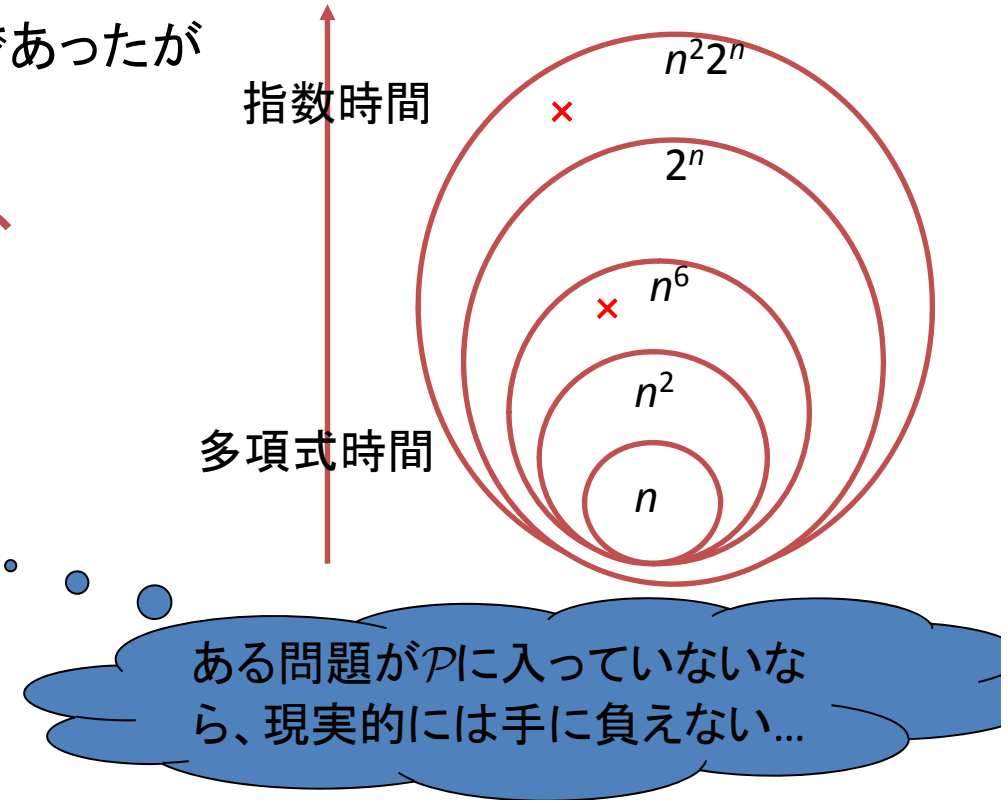
5.2.代表的な計算量クラス

$$\mathcal{P} \equiv \bigcup_{p:\text{多項式}} \mathrm{TIME}(p(l))$$

$$\mathcal{E} \equiv \bigcup_{c>1} \mathrm{TIME}(2^{cl})$$

$$\mathcal{EXP} \equiv \bigcup_{p:\text{多項式}} \mathrm{TIME}(2^{p(l)})$$

$\mathcal{C}$集合： 計算量クラス$\mathcal{C}$に入る集合．
$\mathcal{C}$問題： $\mathcal{C}$集合の認識問題



5. Computational Complexity

5.1. Time Complexity Classes

5.1.3. Time complexity of a problem

Definition: For a function $t(n)$ over natural numbers, the set of all sets (i.e. recognition problems) with time complexities $O(t(n))$ is called **$O(t(n))$-time complexity class**, and it is denoted by TIME($t(n)$). Such a function $t(n)$ is called a time limit.

Ex. 1 PRIME was in TIME($n^2 2^n$),
but now it is in TIME(n^6).

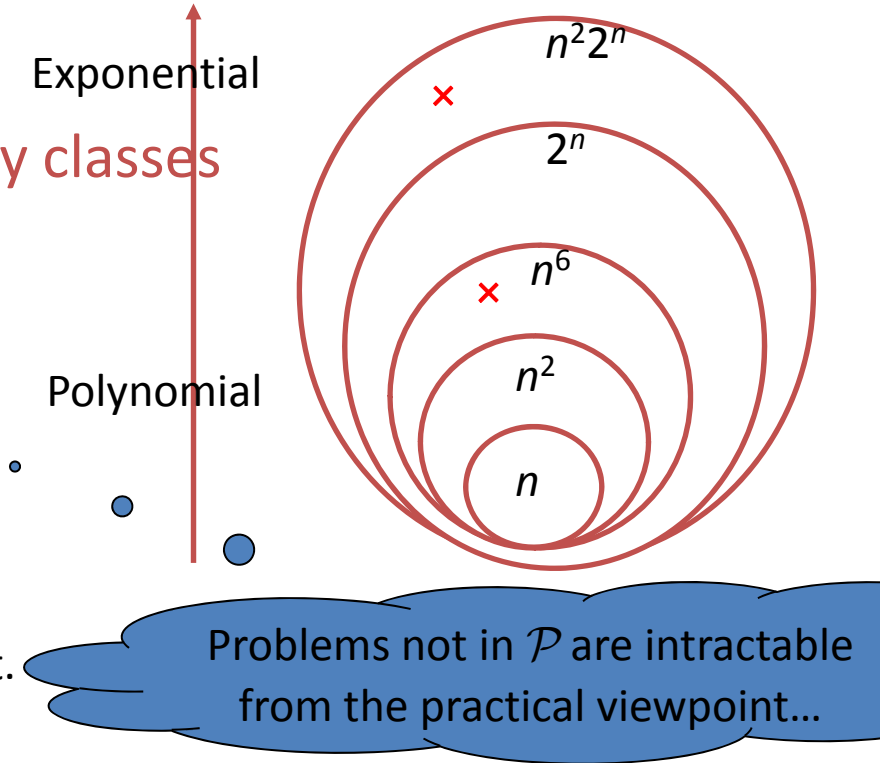
5.2. Representative time complexity classes

$$\mathcal{P} \equiv \bigcup_{p:\text{Polynomial}} \text{TIME}(p(l))$$

$$\mathcal{E} \equiv \bigcup_{c>1} \text{TIME}(2^{cl})$$

$$\mathcal{EXP} \equiv \bigcup_{p:\text{Polynomial}} \text{TIME}(2^{p(l)})$$

$\mathcal{C}$ set : set in the complexity class $\mathcal{C}$.
$\mathcal{C}$ problem : problem of recognizing a $\mathcal{C}$ set.



5.計算量の理論

5.2. 代表的な時間計算量クラス

5.2.2. 代表的な問題とその計算量

5.2.2.1. 命題論理式の評価 (PROP-EVAL)

入力 : $<F, < a_1, a_2, \ldots, a_n >>$

- F は拡張命題論理式
- $(a_1, a_2, \ldots, a_n)$ は F への真偽値割当て

質問 : $F(a_1, a_2, \ldots, a_n) = 1$?

(x,y)	$x \rightarrow y$ $(\neg x \vee y)$	$x \leftrightarrow y$ $((x \rightarrow y) \wedge (y \rightarrow x))$
$(0,0)$	1	1
$(0,1)$	1	0
$(1,0)$	0	0
$(1,1)$	1	1

5. Computational Complexity

5.2. Representative Time Complexity Classes

5.2.2. Representative problems and their complexity

5.2.2.1. Problem of evaluating propositional expression(PROP-EVAL)

Input : $<F, < a_1, a_2, \dots , a_n >>$

- F is an extended propositional expression
- $(a_1, a_2, \dots , a_n)$ is a truth assignment to F

Question : $F(a_1, a_2, \dots , a_n) = 1$?

(x,y)	$x \rightarrow y$ $(\neg x \vee y)$	$x \leftrightarrow y$ $((x \rightarrow y) \wedge (y \rightarrow x))$
$(0,0)$	1	1
$(0,1)$	1	0
$(1,0)$	0	0
$(1,1)$	1	1

5.計算量の理論

5.2.代表的な時間計算量クラス

5.2.2.代表的な問題とその計算量

5.2.2.1.命題論理式の評価(PROP-EVAL)

入力 : $<F, < a_1, a_2, \dots, a_n >>$

- F は拡張命題論理式
- $(a_1, a_2, \dots, a_n)$ は F への真偽値割当て

質問 : $F(a_1, a_2, \dots, a_n) = 1$?

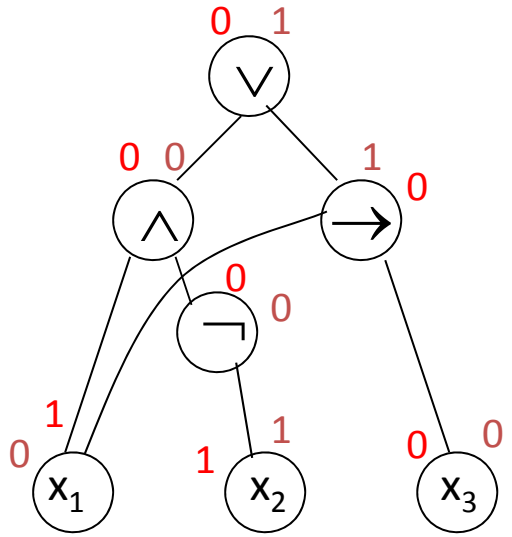
PROP-EVAL ∈ P

F のコードから<u>計算木</u>を作る.
構築に必要な時間は $O(|F|^3)$.
ひとたび計算木ができると,
ボトムアップに計算すると
$F(a_1, a_2, \dots, a_n)$ の値は簡単に求まる.

$F(0,1,0)=1$

$F(1,1,0)=0$

Ex. : $F(x_1, x_2, x_3) = [x_1 \wedge \neg x_2] \vee [x_1 \rightarrow x_3]$



計算木

5. Computational Complexity

5.2. Representative Time Complexity Classes

5.2.2. Representative problems and their complexity

5.2.2.1. Problem of evaluating propositional expression (PROP-EVAL)

Input : $<F, < a_1, a_2, \dots, a_n >>$

- F is an extended prop. expression
- $(a_1, a_2, \dots, a_n)$ is a truth assignment to F

Question : $F(a_1, a_2, \dots, a_n) = 1$?

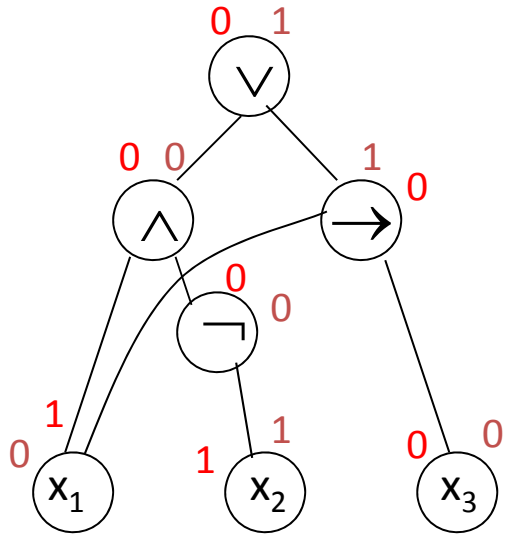
PROP-EVAL $\in$ P

Construct a *computation tree* from a code of F.
It is built in time $O(|F|^3)$.
Once computation tree is built,
we can easily obtain the value
$F(a_1, a_2, \dots, a_n)$ in a bottom-up fashion.

$F(0,1,0)=1$

$F(1,1,0)=0$

Ex. : $F(x_1, x_2, x_3) = [x_1 \wedge \neg x_2] \vee [x_1 \rightarrow x_3]$



computation tree

5.計算量の理論

5.2.代表的な時間計算量クラス

5.2.2.代表的な問題とその計算量

5.2.2.2. 充足可能性 (SAT)

入力 : <*F*> *F* は和積標準形

質問 : $F(a_1, a_2, \ldots, a_n) = 1$とする真偽値割当てが存在するか？

和積標準形 (CNF)

$F = (\bigcirc \vee \bigcirc \vee \ldots \vee \bigcirc) \wedge (\bigcirc \vee \ldots \vee \bigcirc) \wedge \ldots \wedge (\ldots)$

- リテラルの∨の∧で表現される式

k SAT: 各項が *k* 個のリテラルを含む

ちょうど/たかだか

3SAT, 4SATも同様に定義できる.

SAT は任意の CNF を許す

ExSAT は拡張命題論理式（∨, ∧, →, ↔）を許す

5. Computational Complexity

5.2. Representative Time Complexity Classes

5.2.2. Representative problems and their complexity

5.2.2.2. Satisfiability (SAT)

Input : <*F*> *F* is conjunctive normal form

Question : Is there any assignment such that $F(a_1, a_2, \ldots, a_n) = 1$?

Conjunctive Normal Form (CNF)

$F = (\bigcirc \vee \bigcirc \vee \ldots \vee \bigcirc) \wedge (\bigcirc \vee \ldots \vee \bigcirc) \wedge \ldots \wedge (\ldots)$

- described by $\wedge$ of $\vee$ of literals.

k SAT: Each closure contains *k* literals

exactly/at most

We can define 3SAT, 4SAT similarly.

SAT consists of any CNF.

ExSAT consists of any extended propositional expression.

5.計算量の理論

5.2.代表的な時間計算量クラス

5.2.2.代表的な問題とその計算量

5.2.2.3. グラフの到達可能性問題 (ST-CON)

入力 : $\langle G,s,t\rangle$: 無向グラフ G, $1 \leqq s,t \leqq n(=|G|)$

質問 : G は s から t への経路を持つか？

5.2.2.4. オイラー閉路問題 (DEULER)

入力 : $\langle G\rangle$: 有向グラフ G

質問 : G はオイラー閉路を持つか？

実際には,
有向/無向
閉路/パス
という違いは問題にならない

5.2.2.5 ハミルトン閉路問題 (DHAM)

入力 : $\langle G\rangle$: 有向グラフG

質問 : G はハミルトン閉路を持つか？

- 閉路とは両端点を共有する経路.
- オイラー閉路とはすべての辺をちょうど一回通る閉路.
- ハミルトン閉路とはすべての頂点をちょうど一回通る閉路.

5. Computational Complexity

5.2. Representative Time Complexity Classes

5.2.2. Representative problems and their complexity

5.2.2.3. Graph reachability problem (ST-CON)

Input : $\langle G,s,t \rangle$: an undirectd graph G, $1 \leqq s,t \leqq n(=|G|)$

Question : Does G have a path from s to t?

5.2.2.4. Euler cycle problem (DEULER)

Input : $\langle G \rangle$: a directed graph G

Question : Does G have an Euler cycle?

5.2.2.5 Hamiltonian cycle problem (DHAM)

Input : $\langle G \rangle$: a directed graph G

Question : Does G have a Hamiltonian cycle?

Actually,
directed/undirected
cycle/path
do not matter

- Cycle is a path that shares two endpoints.
- Euler cycle is a cycle that visits all edges once.
- Hamiltonian cycle is a cycle that visits all vertices once.

5.計算量の理論

5.2.代表的な時間計算量クラス

5.2.2.代表的な問題とその計算量

以下は既知：

- 以下の問題は P の要素：
  - ✓PROP-EVAL, 2SAT, ST-CON, DEULER
- 以下の問題は $\mathcal{E}$ の要素だが…
  - ✓3SAT, DHAM

P と $\mathcal{E}$ の間(?)にあるクラスNP

5. Computational Complexity

5.2. Representative Time Complexity Classes

5.2.2. Representative problems and their complexity

It is known that :

- The following problems are in P :
  - ✓PROP-EVAL, 2SAT, ST-CON, DEULER
- The following problems are in $\mathcal{E}$, but…
  - ✓3SAT, DHAM

The class NP between P and $\mathcal{E}$?

5.計算量の理論

5.3.クラスNP

5.3.0. 非決定性計算とは

(3SAT, DHAMといった)ある種の問題には，次のような**共通で自然な**性質がある;

- ひとたび解が得られると，その正当性は簡単にチェックできる
- 解を見つけるのは大変そうに思える．可能な場合をしらみつぶしに調べる必要がありそうに見える．

- 現実の自然な問題の多くはこの性質をもつ．
- この性質を表現するのが「非決定性計算」

5. Computational Complexity

5.3. Class NP

5.3.0. Nondeterministic computation

Some problems (like 3SAT, DHAM, etc.) have a *common and natural* property;

- once you get a solution, you can check it efficiently
- without solution, it seems to be quite difficult; you may check all possibilities

• Many natural problems have this property in the real problems.

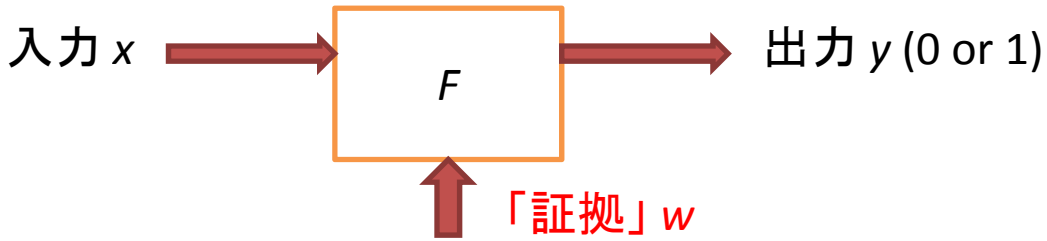
• This property leads us to the notion of "nondeterministic computation"

5.計算量の理論

5.3.クラスNP

5.3.0.非決定性計算とは

- 関数の観点からみると:



以下の関数Fが存在するとき L はNP 集合と呼ばれる:

1. 各xに対して，2進列の「**証拠**」w が存在
2. $|w|$ は $|x|$の多項式で上から抑えられる
3. F は$|x|$と$|w|$の多項式時間でwを使って$x \in L$ を認識する

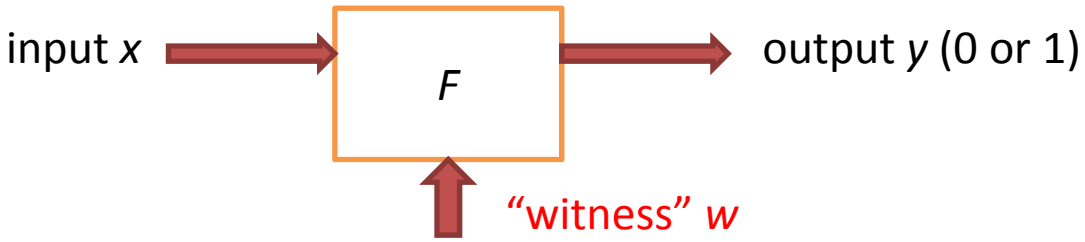
c.f.: NP=**N**ondeterministic **P**olynomial

5. Computational Complexity

5.3. Class NP

5.3.0. Nondeterministic computation

- From the viewpoint of Function:



L is called an NP set if there is a function *F* s.t.

1. For each *x*, there is a binary string "witness" *w* s.t.
2. $|w|$ is bounded by a polynomial of $|x|$
3. *F* recognizes $x \in L$ with *w* in polynomial time of $|x|$ and $|w|$

c.f. : NP=**N**ondeterministic **P**olynomial

5.計算量の理論

5.3.クラスNP

5.3.0.非決定性計算とは

- 命題論理の視点からみると:

この文字列
wを「証拠」
とよぶ

集合Lに対して多項式qと多項式で計算できる述語Rがあり,
以下を満たすとする:

$$\text{for each } x \in \Sigma^*, x \in L \leftrightarrow \exists w \in \Sigma^* : | w | \leq q(| x |)[R(x, w)]$$

つまり,

$$L = \{x : \exists w \in \Sigma^*[| w | \leq q(| x |) \wedge R(x, w)]\}$$

このとき L はNP 集合とよばれ,
Lの認識問題はNP 問題とよばれる.
またNP 集合全体の集合をクラスNPとよぶ.

c.f. : NP=**N**ondeterministic **P**olynomial

5. Computational Complexity

5.3. Class NP

5.3.0. Nondeterministic computation

- From the viewpoint of Logic:

Such a string *w* is called "witness"

Suppose that we have a polynomial q and polynomial time computable predicate R for a set L such that

$$\text{for each } x \in \Sigma^*, x \in L \leftrightarrow \exists w \in \Sigma^* :| w | \leq q(| x |)[R(x, w)]$$

i.e., $$L = \{x : \exists w \in \Sigma^*[\,| w | \leq q(| x |) \wedge R(x, w)]\}$$

Then, L is called an NP set, and the problem of recognizing L is called an NP problem.
Also, the whole set of NP sets is called the class NP.

c.f. : NP=**N**ondeterministic **P**olynomial

5.計算量の理論

5.3.クラスNP

5.3.0.非決定性計算とは

「非決定性選択」はある種の並列計算とみなすこともでき，二つの計算プロセスの生成と考えてもよい．

- チューリングマシンの視点から見ると．

チューリングマシンの「非決定性選択」では，二つの選択肢を「同時に」二つとも選ぶことができる；
つまり「場合(0)と場合(1)のいずれか」という命令がある．

- 非決定性選択は二つの選択肢のうち，「いずれか一方が真」ならば真になる．

このときNP 問題 Lは，非決定性チューリング機械で多項式時間で受理できる問題．

c.f. : NP=**N**ondeterministic **P**olynomial

5. Computational Complexity

5.3. Class NP

5.3.0. Nondeterministic computation

A "nondeterministic choice" is a kind of parallel computing that generates two branches.

- From the viewpoint of Turing Machine:

Suppose that Turing machine has "nondeterministic choice" that admits us to two possible choices at the same time; i.e., it has "one of two cases (0) and (1)" statement.

- A nondeterministic choice allows to assume of two choices and it will be "*true*" if "*at least one of them is true*".

Then, NP problem L can be recognized by a nondeterministic Turing machine in polynomial time.

c.f. : NP=**N**ondeterministic **P**olynomial

5.計算量の理論

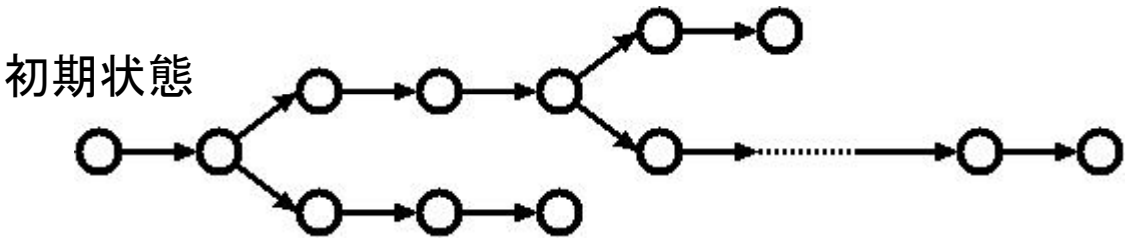
5.3.クラスNP

5.3.0.非決定性計算とは

- チューリングマシンの計算木の観点からみると:

- 決定性のチューリングマシンの計算木はパス(一本道);

- 非決定性のチューリングマシンの計算木は木;



- 各計算プロセスは受理/拒否状態になるか無限ループ
- 木が多項式長の範囲で**受理状態を一つでももてば**受理.

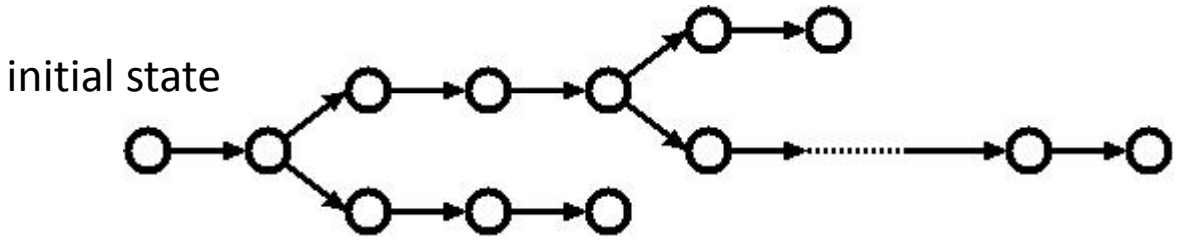
5. Computational Complexity

5.3. Class NP

5.3.0. Nondeterministic computation

- From the viewpoint of the computation tree of a Turing Machine:
  - Computation tree of a deterministic Turing machine forms a path;

  - Computation tree of a *nondeterministic* Turing machine forms a *tree*;



    - each computation halts in an accept/reject state or loop.
    - it accepts if the tree has **at least one "accept"** in poly-length.

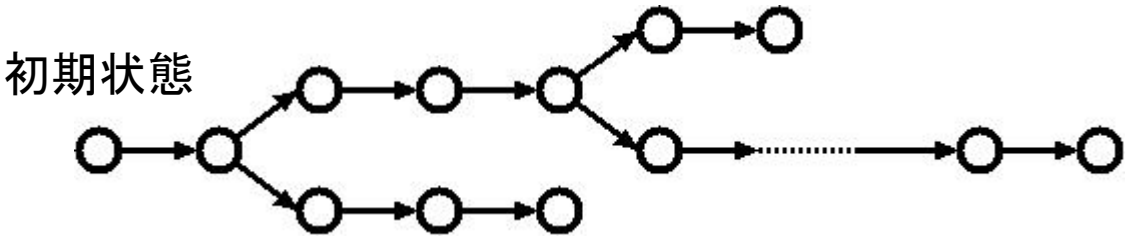
5.計算量の理論

5.3.クラスNP

5.3.0.非決定性計算とは

証拠wは正しい選択肢の列を与える

- チューリングマシンの計算木の観点からみると:
  - 非決定性のチューリングマシンの計算木は木;



- 各計算プロセスは受理/拒否状態になるか無限ループ
- 木が多項式長の範囲で受理状態を一つでももてば受理.

NP 問題 L とは非決定性チューリング機械で
多項式時間で認識できる言語. つまり, 受理状態に至る
n の多項式長の計算パスが存在すればよい.

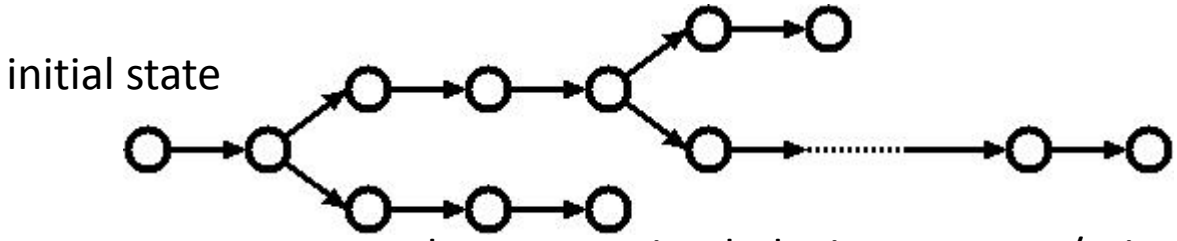
5. Computational Complexity

5.3. Class NP

5.3.0. Nondeterministic computation

The *witness w* gives the right choices

- From the viewpoint of the computation tree of a Turing Machine:
  - Computation tree of a *nondeterministic* Turing machine forms a *tree*;



  - each computation halts in an accept/reject state
  - it accepts if the tree has **at least one "accept"** in poly-length.

An NP problem L is recognized by a nondeterministic Turing machine in polynomial time. That is, there is a computation path to an accept state of length polynomial of *n*.

5.計算量の理論

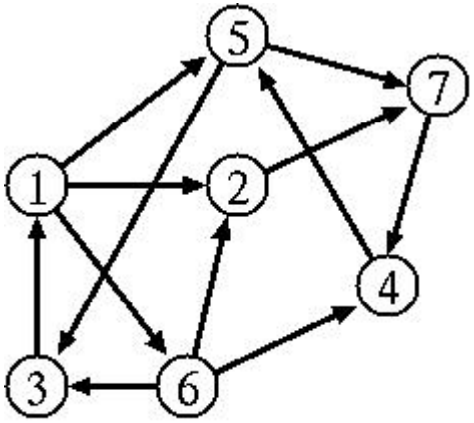
5.3.クラスNP

5.3.1. 代表的なNP問題

- ハミルトン閉路問題 (DHAM)

  入力 : <*G*>: 有向グラフ*G*

  質問 : *G* はハミルトン閉路をもつか?



- 原理的には*n*の順列をすべて試せばよいが，可能な組合せの数は最大で$n!$～n^n
  …指数時間かかってしまう.
- もし *G* がハミルトン閉路Cをもつならこれを証拠にすれば，効率よくそれをチェックすることができる.

5. Computational Complexity

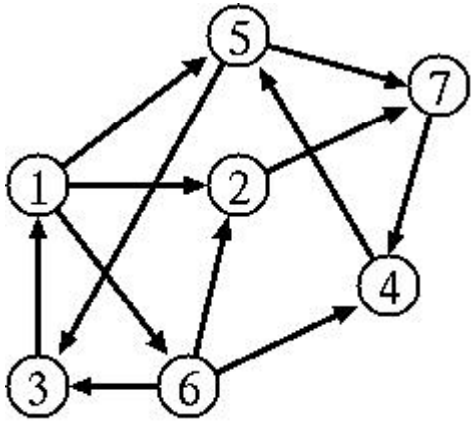
5.3. Class NP

5.3.1. Representative NP problems

- Hamiltonian cycle problem (DHAM)

  Input : <*G*>: a directed graph *G*

  Question : Does *G* have a Hamiltonian cycle?



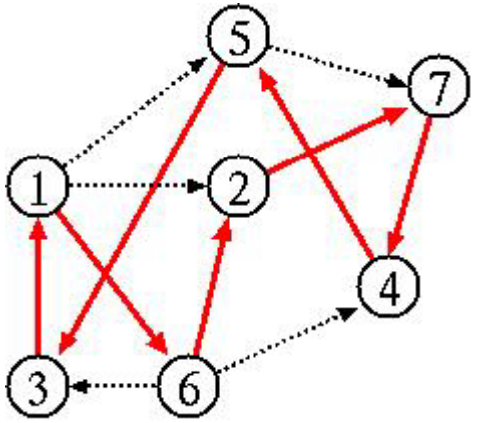
- We can certainly check all possible permutations of *n*, that counts up to $n! \sim n^n$…
  it takes exponential time.
- If *G* has a Hamiltonian cycle *C*, and we have it as a witness, we can check that it surely a Hamiltonian cycle.

1,6,2,7,4,5,3,1

5.計算量の理論

5.3.クラスNP

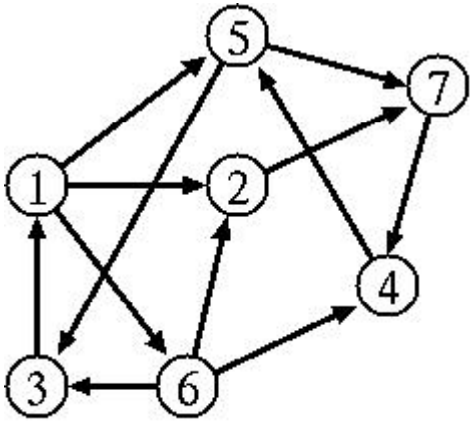
5.3.1.代表的なNP問題



- ハミルトン閉路問題 (DHAM)

  入力 : <*G*>: 有向グラフ*G*

  質問 : *G* はハミルトン閉路をもつか?

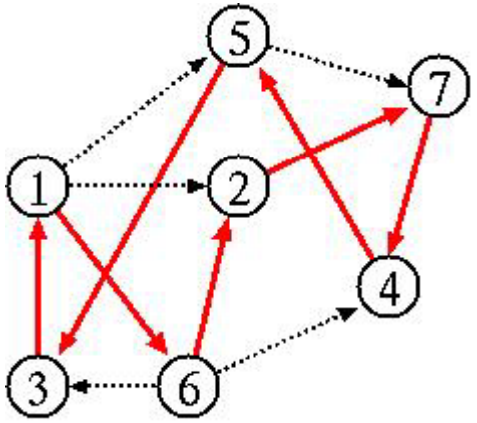


- 原理的には*n*の順列をすべて試せばよいが，可能な組合せの数は最大で$n!$～n^n…
  指数時間かかってしまう.
- もし *G* がハミルトン閉路Cをもつならこれを証拠にすれば，効率よくそれをチェックすることができる.

5. Computational Complexity

5.3. Class NP

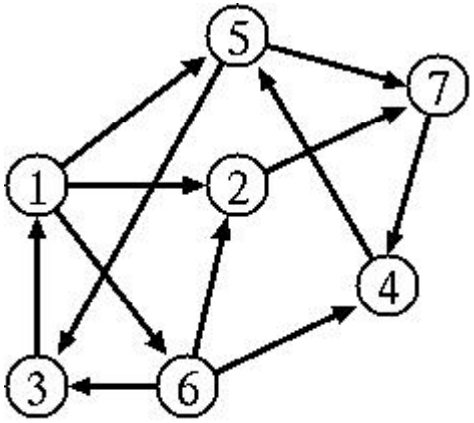
5.3.1. Representative NP problems



- Hamiltonian cycle problem (DHAM)

  Input : <*G*>: a directed graph *G*

  Question : Does *G* have a Hamiltonian cycle?



- We can certainly check all possible permutations of *n*, that counts up to $n! \sim n^n$…
  it takes exponential time.
- If *G* has a Hamiltonian cycle *C*, and we have it as a witness, we can check that it surely a Hamiltonian cycle.

5.計算量の理論

5.3.クラスNP

5.3.1.代表的なNP問題

- SAT, *k*SAT, ExSAT (充足可能性)

  入力 : <*F*> *F* は和積標準形命題論理式

  質問 : $F(a_1, a_2, \ldots, a_n) = 1$となる割当ては存在?

- *F* を充足する割当て *A* があるなら,
  それを証拠として使い, PROP_EVALのときと同じ方法で
  多項式時間でチェックできる.

- もちろん$(a_1, a_2, \ldots, a_n)$のすべての可能な割当てを
  チェックすることはできるが, 可能な割当ての個数は
  2^n なので, 指数時間かかる.

5. Computational Complexity

5.3. Class NP

5.3.1. Representative NP problems

- SAT, *k*SAT, ExSAT (Satisfiability)

  Input : <*F*> *F* is conjunctive normal form

  Question : Any assignment s. t. $F(a_1, a_2, \ldots, a_n) = 1$?

- If *F* is satisfiable by an assignment *A*, and we have it as a witness, we can check it in polynomial time by the same way as the PROP_EVAL.

- We can certainly check all possible assignments of $(a_1, a_2, \ldots, a_n)$. The assignments are 2^n, that takes exponential time.

5.計算量の理論

5.3.クラスNP

5.3.2. NP問題を別の視点から見る

- NP 集合であることの意味は?
  - •命題述語論理によるNP 集合の特徴付けで出てきた q と R を使うと,「$x \in L$?」という質問に次のアルゴリズムで答えることができる.

```
for each w ∈ Σ^{≤q(|x|)} do
  if R(x, w) then accept end-if
end-for;
reject;
```

長さ高々$q(|x|)$のすべての文字列を辞書式に列挙してチェックすれば, 受理または拒否を判断できる.
ただし, こうした文字列は$2^{q(|x|)}$ (指数関数的) 通りある.
こうしたアルゴリズムで認識できる集合をNP集合と考えてもよい.

5. Computational Complexity

5.3. Class NP

5.3.2. Another aspect of the NP problems

- What does it mean by being an NP set?
  - Using q and R satisfying the predicate characterizing an NP set, we can determine "$x \in L$?" in the following way.

```
for each w ∈ Σ^{≤q(|x|)} do
  if R(x, w) then accept end-if
end-for;
reject;
```

If we enumerate and check all possible strings of length at most $q(|x|)$, we can accept or reject them.
Here note that there are $2^{q(|x|)}$ (exponentially many) such strings.

We may think that those sets recognizable as above are NP sets.

5.計算量の理論

5.3. クラスNP

5.3.3. 代表的なNP問題再び

- **ナップサック問題 (KNAP)**

  入力: 自然数のn+1個組$< a_1, a_2, \dots, a_n, b >$

  質問: 添え字の集合$S \subseteq \{1, \dots, n\}$で$\sum_{i \in S} a_i = b$ を満たすものはあるか?

- **ビン詰め問題 (BIN)**

  入力: 自然数のn+2個組$< a_1, a_2, \dots, a_n, b, k >$

  質問: 添え字の集合 $U=\{1, \dots, n\}$ の分割$U_1, \dots, U_k$で $\sum_{i \in U_j} a_i \leq b$ を満たすものはあるか?

- **頂点被覆問題 (VC)**

  入力: 無向グラフGと自然数kの組 $<G, k>$

  質問: G上に大きさ k の頂点被覆は存在するか?

> 頂点被覆 Sとは，各辺$\{u,v\}$に対してu,vの少なくともどちらか一方をふくむ頂点集合

5. Computational Complexity

5.3. Class NP

5.3.3. More representative NP problems

- Knapsack Problem (KNAP)

  Input : n+1 tuple of natural numbers $< a_1, a_2, \dots, a_n, b>$

  Question: Is there a set of indices $S \subseteq \{1, \dots, n\}$ s.t. $\sum_{i \in S} a_i = b$?

- Bin Packing Problem (BIN)

  Input : n+2 tuple of natural numbers $< a_1, a_2, \dots, a_n, b, k>$

  Question: Is there a partition of a set of indices $U=\{1, \dots, n\}$ into $U_1, \dots, U_k$ such that $\sum_{i \in U_j} a_i \leq b$ for each j?

- Vertex Cover Problem (VC)

  Input : pair of undirected graph G and natural number k $<G, k>$

  Question: Is there a vertex cover of k vertices over G?

Vertex Cover S contains at least one of u and v for each edge $\{u,v\}$.

5.計算量の理論

5.4. クラスcoNP

定義

集合 L が coNP に属する必要十分条件は,
その補集合がNPに属すること.

定理

任意の集合 L に対して, 以下の二つは同値である.

(a) $L \in$ coNP

(b) Lは多項式qと多項式時間で計算できる述語Qを使って
次のように書ける: $L = \{x : \forall w \in \Sigma^* : \ |w| \leq q(|x|)[Q(x,w)]\}$

[注意] coP はPと同値であることがすぐにわかるので,
定義しても無意味.

5. Computational Complexity

5.4. Class coNP

Definition

A set *L* is in coNP if and only if its complement belongs to NP.

Theorem

For every set *L*, the following conditions are equivalent.

(a) $L \in$ coNP

(b) The set *L* can be represented as

$$L = \{x : \forall w \in \Sigma^* : \ \ | w | \leq q(| x |)[Q(x, w)]\}$$

by using some polynomial *q* and polynomial-time computable predicate *Q*.

[Note] It is nonsense to define coP since it is equal to P.

5.計算量の理論

5.5.計算量クラスの関係

定理 $P \subseteq E \subseteq EXP$

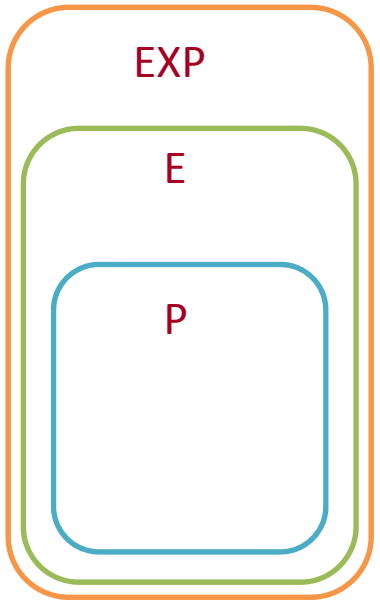
証明: 定義より明らか.

定理 $P \subsetneqq E \subsetneqq EXP$

証明: 本書の範囲を超えるので省略.
(アイデアの概略: 対角線論法を巧妙に使うと, 例えば$t_1(n)^3=O(t_2(n))$といった関数に対して次の階層定理を示すことができる.

$\mathrm{TIME}(t_1(n)) \subsetneqq \mathrm{TIME}(t_2(n))$

*真に異なる*階層構造が成立する



5. Computational Complexity

5.5. Relations in the Complexity Classes

Theorem $P \subseteq E \subseteq EXP$

Proof: Obvious from the definition.

Theorem $P \subsetneq E \subsetneq EXP$

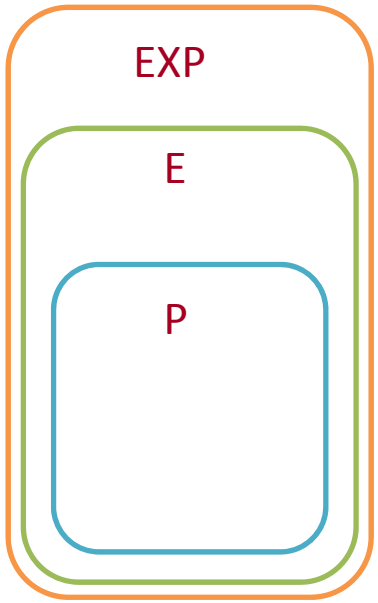
Proof: Out of scope in this class…

(Brief idea: We can use *diagonalization* to show a hierarchy theorem that says

$\text{TIME}(t_1(n)) \subsetneq \text{TIME}(t_2(n))$

for, e.g., $t_1(n)^3 = O(t_2(n))$).

We have a *proper* hierarchy



5.計算量の理論

5.5.計算量クラスの関係

定理
(1)P ⊆ NP, P ⊆coNP (∴P ⊆ NP ∩ coNP)
(2)NP ⊆ EXP, coNP ⊆ EXP (∴ NP ∪ coNP ⊆ EXP)

証明(概略):
(1) P⊆NP (P⊆coNP も同様)
NPの定義の中の「証拠」を無視すれば,
Pの定義と同値なものが得られる.
(2) NP⊆EXP (coNP⊆EXP も同様)
長さmのすべての文字列に対して
それが長さmの「証拠」wになるかどうかを
指数時間かけてチェックすればよい.

5. Computational Complexity

5.5. Relations in the Complexity Classes

Theorem

(1) $P \subseteq NP$, $P \subseteq coNP$ ($\therefore P \subseteq NP \cap coNP$)

(2) $NP \subseteq EXP$, $coNP \subseteq EXP$ ($\therefore NP \cup coNP \subseteq EXP$)

Proof (Outline):

(1) $P \subseteq NP$ ($P \subseteq coNP$ is similar)

Ignoring the "witness" in the definition of NP, we immediately obtain the definition of P.

(2) $NP \subseteq EXP$ ($coNP \subseteq EXP$ is similar)

For the "witness" *w* of length *m*, we can check all possible strings of length *m* in exponential time.

5.計算量の理論

5.5.計算量クラスの関係

定理
(1) NP ⊆ coNP → NP = coNP
(2) coNP ⊆ NP → NP = coNP
(3) NP ≠ coNP → P ≠ NP

注: (3)よりNP ≠ co-NP の証明はP ≠ NPの証明よりも難しい.

証明 :

(1) NP ⊆ coNP → NP = coNP

仮定より coNP ⊆ NPを示せばよい.

そこで任意の $L \in$ coNPに対して $L \in$ NP を示す.

$L \in$ coNP $\Leftrightarrow \overline{L} \in$ NP (定義より)

$\rightarrow \overline{L} \in$ coNP (NP ⊆ co-NP)

$\Leftrightarrow L \in$ NP (定義と $L=\overline{\overline{L}}$ より)

5. Computational Complexity

5.5. Relations in the Complexity Classes

Theorem

(1) NP $\subseteq$ coNP $\rightarrow$ NP = coNP

(2) coNP $\subseteq$ NP $\rightarrow$ NP = coNP

(3) NP $\neq$ coNP $\rightarrow$ P $\neq$ NP

Note: From (3), proof for NP $\neq$ co-NP is harder than that for P $\neq$ NP.

Proof :

(1) NP $\subseteq$ coNP $\rightarrow$ NP = coNP

By assumption, it is sufficient to show that coNP $\subseteq$ NP.

We will prove $L \in$ NP for any $L \in$ coNP.

$L \in$ coNP $\Leftrightarrow \overline{L} \in$ NP (by Definition)

$\rightarrow \overline{L} \in$ coNP (NP $\subseteq$ co-NP)

$\Leftrightarrow L \in$ NP (Definition and $L=\overline{\overline{L}}$)

5.計算量の理論

5.5.計算量クラスの関係

定理

(1) NP ⊆ coNP → NP = coNP

(2) coNP ⊆ NP → NP = coNP

(3) NP ≠ coNP → P ≠ NP

注: (3)よりNP ≠ co-NP の証明はP ≠ NPの証明よりも難しい.

証明: (3) NP ≠ coNP → P ≠ NP

以下の対偶を示す: P = NP → NP = coNP

P=NPと仮定すると, 任意の集合 L に対して以下を得る

$L \in \mathrm{NP} \Leftrightarrow L \in \mathrm{P}$ (P = NP)

$\Leftrightarrow \bar{L} \in \mathrm{P}$ (P = coP)

$\Leftrightarrow \bar{L} \in \mathrm{NP}$ (P = NP)

$\Leftrightarrow L\ (=\bar{\bar{L}}) \in \mathrm{coNP}$ (NP/coNPの定義より)

∴ NP = coNP Q.E.D.

5. Computational Complexity

5.5. Relations in the Complexity Classes

Theorem

(1) NP $\subseteq$ coNP $\rightarrow$ NP = coNP

(2) coNP $\subseteq$ NP $\rightarrow$ NP = coNP

(3) NP $\neq$ coNP $\rightarrow$ P $\neq$ NP

Note: From (3), proof for NP $\neq$ co-NP is harder than that for P $\neq$ NP.

Proof: (3) NP $\neq$ coNP $\rightarrow$ P $\neq$ NP

Contraposition : P = NP $\rightarrow$ NP = coNP

If we assume P=NP, for any L we have

$L \in \text{NP} \Leftrightarrow L \in \text{P}$ (P = NP)

$\Leftrightarrow \bar{L} \in \text{P}$ (P = coP)

$\Leftrightarrow \bar{L} \in \text{NP}$ (P = NP)

$\Leftrightarrow L\ (=\bar{\bar{L}}) \in \text{coNP}$ (Definitions of NP/coNP)

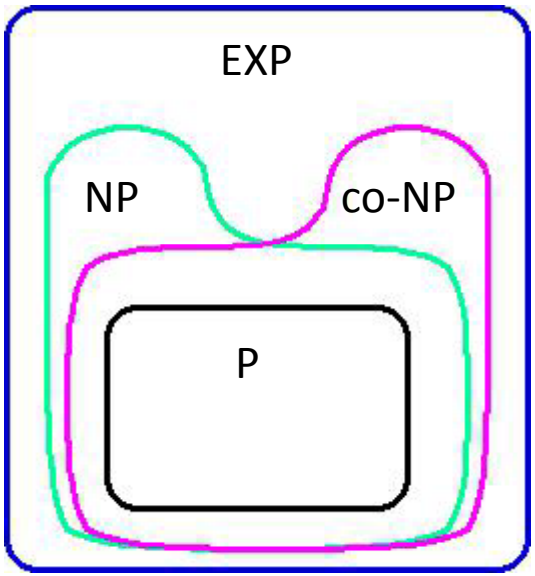
$\therefore$ NP = coNP Q.E.D.

5.計算量の理論

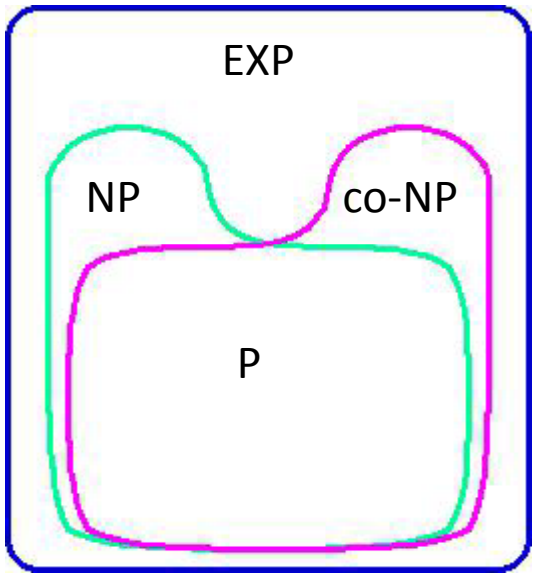
5.5. 計算量クラスの関係

定理
(1) NP ⊆ coNP → NP = coNP
(2) coNP ⊆ NP → NP = coNP
(3) NP ≠ coNP → P ≠ NP

P≠ NPが成立すると強く信じられているので,
以下の構造になっていると予想される.



または



5. Computational Complexity

5.5. Relations in the Complexity Classes

Theorem

(1) NP ⊆ coNP → NP = coNP

(2) coNP ⊆ NP → NP = coNP

(3) NP ≠ coNP → P ≠ NP

We strongly believe that P ≠ NP, and then we have