精読『アルゴリズムイントロダクション 第4版』

第2回

Panot

最終更新：April 29, 2023

復習

- アルゴリズムは明確に定義された手続き。
- アルゴリズムによって効率が異なり、必要なリソースも大きく変わる。
- アルゴリズムの実行時間を比較する際、増加が一番速い項で特徴づけられる。

今日の内容

- 分割統治、再帰、漸化式
- アルゴリズム的な漸化式の解く方法
- 確率的解析と乱択アルゴリズム

Section 1

分割統治、再帰、漸化式

基底ケースと再帰ケース

問題のサイズが十分に小さい場合は**基底ケース（base case）**で、直接的な方法で解く。
そうでない場合は**再帰ケース（recursive case）**で、このように問題を部分問題に分割し、統合して解を得る。

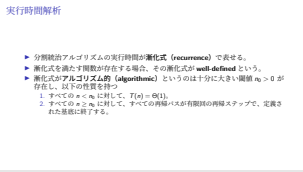
分割 問題をいくつかの同じ問題のより小さいインスタンスである部分問題に分割。

統治 部分問題を再帰的に解く。

統合 部分問題の解を組み合わせて元の問題の解を得る。

実行時間解析

- 分割統治アルゴリズムの実行時間が**漸化式（recurrence）**で表せる。
- 漸化式を満たす関数が存在する場合、その漸化式が **well-defined** という。
- 漸化式が**アルゴリズム的（algorithmic）**というのは十分に大きい閾値 $n_0 > 0$ が存在し、以下の性質を持つ
  1. すべての $n < n_0$ に対して、$T(n) = \Theta(1)$。
  2. すべての $n \geq n_0$ に対して、すべての再帰パスが有限回の再帰ステップで、定義された基底に終了する。



- アルゴリズム的とは要するに、ある定義された手続きによって条件下のあらゆる入力に対して有限時間に終了する

分割統治と漸化式

- この章は正方行列の積を計算問題として、いくつかのアルゴリズムを紹介して分析する。
- $n \times n$ の正方行列の積は直接計算した場合、実行時間 $\Theta(n^3)$ となる。
- 単純な分割統治法で 4 つの大きさ $n/2$ の部分問題にして計算したら、実行時間は漸化式 $T(n) = 8T(n/2) + \Theta(1)$ になるが、この実行時間は $\Theta(n^3)$ である。
- Strassen の分割統治法を使うと、実行時間は $T(n) = 7T(n/2) + \Theta(n^2) = \Theta(n^{\lg 7}) = \Theta(n^{2.81})$ である。

- TODO: describe above

分割統治と漸化式

- この章は正方行列の積を計算問題として、いくつかのアルゴリズムを紹介して分析する。
- $n \times n$ の正方行列の積は直接計算した場合、実行時間 $\Theta(n^3)$ となる。
- 単純な分割統治法で 4 つの大きさ $n/2$ の部分問題にして計算したら、実行時間は漸化式 $T(n) = 8T(n/2) + \Theta(1)$ になるが、この実行時間は $\Theta(n^3)$ である。
- Strassen の分割統治法を使うと、実行時間は $T(n) = 7T(n/2) + \Theta(n^2) = \Theta(n^{\log 7}) = \Theta(n^{2.81})$ である。

漸化式を解く方法

- **置き換え法（substitution method）**では、推測し数学的帰納法で証明する。
- **再帰木法（recursion-tree method）**では、漸化式を木と見立てて、各階層の総実行時間を計算し、全ての階層の総和を計算する。
- **マスター法（master method）**では、$a > 0$ かつ $b > 1$ のときの漸化式 $T(n) = aT(n/b) + f(n)$ の閉じた限界式を求める。
- **Akra-Bazzi 法**では、分割統治の漸化式の閉じた限界式を一般的に求める。

- TODO: describe each

漸化式を解く方法

- **置き換え法（substitution method）**では、推測し数学的帰納法で証明する。
- **再帰木法（recursion-tree method）**では、漸化式を木と見立てて、各階層の総実行時間を計算し、全ての階層の総和を計算する。
- **マスター法（master method）**では、$a > 0$ かつ $b > 1$ のときの漸化式 $T(n) = aT(n/b) + f(n)$ の閉じた限界式を求める。
- **Akra-Bazzi 法**では、分割統治の漸化式の閉じた限界式を一般的に求める。

正方行列の積

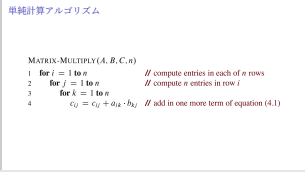
$n \times n$ の正方行列 $A = (a_{ik})$ と $B = (b_k j)$ に対して、積 $C = A \cdot B$ も $n \times n$ の正方行列で C の要素 c_{ij} は以下のように成り立つ。

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

単純計算アルゴリズム

MATRIX-MULTIPLY(A, B, C, n)

```
1  for i = 1 to n                    // compute entries in each of n rows
2      for j = 1 to n                // compute n entries in row i
3          for k = 1 to n
4              c_ij = c_ij + a_ik · b_kj   // add in one more term of equation (4.1)
```



- TODO: Explain above algorithm

単純な分割統治法

行列 A と B を 4 分割して、以下の計算をする

$$\begin{aligned}\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} &= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \\ &= \begin{pmatrix} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{pmatrix}\end{aligned}$$

単純な分割統治法のアルゴリズム

```
MATRIX-MULTIPLY-RECURSIVE(A, B, C, n)
 1  if n == 1
 2      // Base case.
 3      c11 = c11 + a11 · b11
 4      return
 5  // Divide.
 6  partition A, B, and C into n/2 × n/2 submatrices
        A11, A12, A21, A22; B11, B12, B21, B22;
        and C11, C12, C21, C22; respectively
 7  // Conquer.
 8  MATRIX-MULTIPLY-RECURSIVE(A11, B11, C11, n/2)
 9  MATRIX-MULTIPLY-RECURSIVE(A11, B12, C12, n/2)
10  MATRIX-MULTIPLY-RECURSIVE(A21, B11, C21, n/2)
11  MATRIX-MULTIPLY-RECURSIVE(A21, B12, C22, n/2)
12  MATRIX-MULTIPLY-RECURSIVE(A12, B21, C11, n/2)
13  MATRIX-MULTIPLY-RECURSIVE(A12, B22, C12, n/2)
14  MATRIX-MULTIPLY-RECURSIVE(A22, B21, C21, n/2)
15  MATRIX-MULTIPLY-RECURSIVE(A22, B22, C22, n/2)
```

実行時間は漸化式 $T(n) = 8T(n/2) + \Theta(1)$ になるが、この実行時間は $\Theta(n^3)$ である。

Strassen のアルゴリズム

- 詳細は各自確認してください。
- 要は実行時間は $T(n) = 7T(n/2) + \Theta(n^2) = \Theta(n^{2.81})$。

Section 2

アルゴリズム的な漸化式の解く方法

- **置き換え法（substitution method）**では、推測し数学的帰納法で証明する。
- **再帰木法（recursion-tree method）**では、漸化式を木と見立てて、各階層の総実行時間を計算し、全ての階層の総和を計算する。
- **マスター法（master method）**では、$a > 0$ かつ $b > 1$ のときの漸化式 $T(n) = aT(n/b) + f(n)$ の閉じた限界式を求める。
- **Akra-Bazzi 法**では、分割統治の漸化式の閉じた限界式を一般的に求める。

置き換え法（substitution method）

- 定数を変数のままにして、限界を推測する。
- 数学的帰納法を用いて限界の正しさを証明し、定数を特定する。

例

$$T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n) \tag{1}$$

のとき、$T(n) = O(n \lg n)$ を証明したい。

すべての $n \geq n_0$ に対して $T(n) \leq cn \lg n$、と仮定する。
ただし、定数 $c > 0$ と $n_0 > 0$ は過程で求める。

これが証明できれば、$T(n) = O(n \lg n)$ も証明されることになる。

例：数学的帰納法の帰納ケース（inductive case）

- $[n_0, n)$ で $T(n) \leq cn \lg n$ が成り立つなら、n でも成り立つことを証明する。

例：数学的帰納法の帰納ケース（inductive case）

- $[n_0, n)$ で $T(n) \leq cn \lg n$ が成り立つなら、n でも成り立つことを証明する。
- $n \geq 2n_0$ なら、$\lfloor n/2 \rfloor$ で成り立つ。従って、$T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$。

例：数学的帰納法の帰納ケース（inductive case）

- $[n_0, n)$ で $T(n) \leq cn \lg n$ が成り立つなら、n でも成り立つことを証明する。
- $n \geq 2n_0$ なら、$\lfloor n/2 \rfloor$ で成り立つ。従って、$T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$。
- （1）に代入すると、

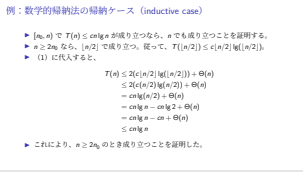
$$
\begin{aligned}
T(n) &\leq 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + \Theta(n) \\
&\leq 2(c(n/2) \lg(n/2)) + \Theta(n) \\
&= cn \lg(n/2) + \Theta(n) \\
&= cn \lg n - cn \lg 2 + \Theta(n) \\
&= cn \lg n - cn + \Theta(n) \\
&\leq cn \lg n
\end{aligned}
$$

例：数学的帰納法の帰納ケース（inductive case）

- $[n_0, n)$ で $T(n) \leq cn \lg n$ が成り立つなら、n でも成り立つことを証明する。
- $n \geq 2n_0$ なら、$\lfloor n/2 \rfloor$ で成り立つ。従って、$T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$。
- （1）に代入すると、

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + \Theta(n) \\ &\leq 2(c(n/2) \lg(n/2)) + \Theta(n) \\ &= cn \lg(n/2) + \Theta(n) \\ &= cn \lg n - cn \lg 2 + \Theta(n) \\ &= cn \lg n - cn + \Theta(n) \\ &\leq cn \lg n \end{aligned}$$

- これにより、$n \geq 2n_0$ のとき成り立つことを証明した。



- 最後の行では c の選び方によって、cn が $\Theta(n)$ に隠されている無名の関数より大きくなるために不等式が成り立つ。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。
- ここで c と n_0 の条件も求める。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。
- ここで c と n_0 の条件も求める。
- $n_0 > 1$ の条件下 $\lg n > 0$ よって $n \lg n > 0$。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。
- ここで c と n_0 の条件も求める。
- $n_0 > 1$ の条件下 $\lg n > 0$ よって $n \lg n > 0$。
- $n_0 = 2$ と選ぶ。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。
- ここで c と n_0 の条件も求める。
- $n_0 > 1$ の条件下 $\lg n > 0$ よって $n \lg n > 0$。
- $n_0 = 2$ と選ぶ。
- $T(n)$ はアルゴリズム的という前提の元で $T(2)$ と $T(3)$ が定数。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。
- ここで c と n_0 の条件も求める。
- $n_0 > 1$ の条件下 $\lg n > 0$ よって $n \lg n > 0$。
- $n_0 = 2$ と選ぶ。
- $T(n)$ はアルゴリズム的という前提の元で $T(2)$ と $T(3)$ が定数。
- $c = \max\{T(2), T(3)\}$ と選ぶ。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。
- ここで c と n_0 の条件も求める。
- $n_0 > 1$ の条件下 $\lg n > 0$ よって $n \lg n > 0$。
- $n_0 = 2$ と選ぶ。
- $T(n)$ はアルゴリズム的という前提の元で $T(2)$ と $T(3)$ が定数。
- $c = \max\{T(2), T(3)\}$ と選ぶ。
- よって、

$$T(2) \leq c < (2 \lg 2)c$$
$$T(3) \leq c < (3 \lg 3)c$$

基底ケースを証明することができた。

例：数学的帰納法の基底ケース（base case）

- $n_0 \leq n < 2n_0$ のとき、$T(n) \leq cn \lg n$ が成り立つことを証明する。
- ここで c と n_0 の条件も求める。
- $n_0 > 1$ の条件下 $\lg n > 0$ よって $n \lg n > 0$。
- $n_0 = 2$ と選ぶ。
- $T(n)$ はアルゴリズム的という前提の元で $T(2)$ と $T(3)$ が定数。
- $c = \max\{T(2), T(3)\}$ と選ぶ。
- よって、

$$T(2) \leq c < (2 \lg 2)c$$
$$T(3) \leq c < (3 \lg 3)c$$

  基底ケースを証明することができた。
- 従って、すべての $n \geq 2$ に対して、$T(n) \leq cn \lg n$ の証明ができた。

練習問題

手を動かして証明してみましょう！

1. $T(n) = T(n-1) + n$ の解は $T(n) = O(n^2)$
2. $T(n) = T(n/2) + \Theta(1)$ の解は $T(n) = O(\lg n)$
3. $T(n) = 2T(n/2) + n$ の解は $T(n) = \Theta(n \lg n)$
4. $T(n) = 2T(n/2 + 17) + n$ の解は $T(n) = O(n \lg n)$
5. $T(n) = 2T(n/3) + \Theta(n)$ の解は $T(n) = \Theta(n)$
6. $T(n) = 4T(n/2) + \Theta(n)$ の解は $T(n) = \Theta(n^2)$

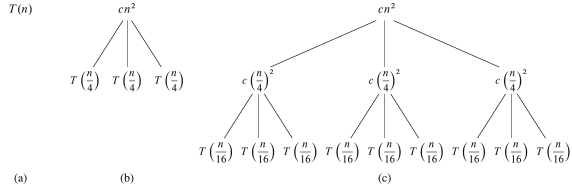
再帰木法（recursion-tree method）

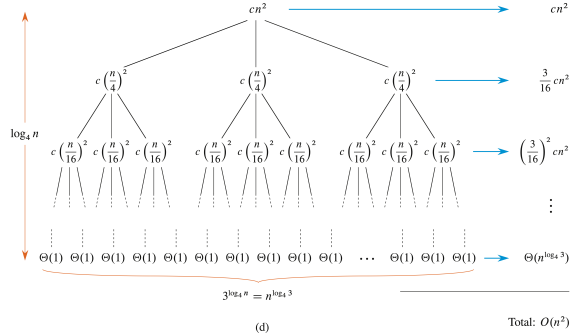
- 漸化式を木と見立てて、各階層の総実行時間を計算し、全ての階層の総和を計算する。
- この方法で直接照明してもいいし、置き換え法の推測のために使ってもいい。

例

$T(n) = 3T(n/4) + \Theta(n^2)$ の上界を求めたい。

例：図


$T(n)$
cn^2
$T\left(\frac{n}{4}\right)$ $T\left(\frac{n}{4}\right)$ $T\left(\frac{n}{4}\right)$
$c\left(\frac{n}{4}\right)^2$
$T\left(\frac{n}{16}\right)$
(a)
(b)
(c)


cn^2
$\frac{3}{16}cn^2$
$\left(\frac{3}{16}\right)^2 cn^2$
$\log_4 n$
$\Theta(1)$
$\Theta(n^{\log_4 3})$
$3^{\log_4 n} = n^{\log_4 3}$
(d)
Total: $O(n^2)$

例：計算

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2)
\end{aligned}
$$

マスター法（master method）

- $a > 0$ かつ $b > 1$ のときの漸化式 $T(n) = aT(n/b) + f(n)$ の閉じた限界式を求める。
- この漸化式は問題の大きさ n を a 個大きさ n/b の部分問題に分割統治法を用いたアルゴリズムの実行時間を示す。$f(n)$ は分割と統合にかかるコストを表す。

Theorem (マスター定理（master theorem）)

$a > 0$ と $b > 1$ を定数として、$f(n)$ を十分に大きい実数に定義されかつ非負の関数とする。$n \in \mathbb{N}$ 上の漸化式 $T(n)$ をこのように定義する：

$$T(n) = aT(n/b) + f(n)$$

ただし $aT(n/b)$ は厳密に $a'T(\lfloor n/b \rfloor) + a''T(\lceil n/b \rceil)$ であり、$a' \geq 0$ かつ $a'' \geq 0$ かつ $a = a' + a''$ である。このとき、$T(n)$ の漸近的特徴は以下のように得られる。

1. $f(n) = O(n^{\log_b a - \epsilon})$ となるような定数 $\epsilon > 0$ が存在したら、$T(n) = \Theta(n^{\log_b a})$ である。
2. $f(n) = \Theta(n^{\log_b a} \lg^k n)$ となるような定数 $k \geq 0$ が存在したら、$T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$ である。
3. $f(n) = \Omega(n^{\log_b a + \epsilon})$ となるような定数 $\epsilon > 0$ が存在し、しかもある定数 $c < 1$ と十分に大きい n に対して、$af(n/b) \leq cf(n)$ が成り立ったら、$T(n) = \Theta(f(n))$ である。

マスター定理を直感的に見てみよう

- 再帰による $aT(n/b)$ と $f(n)$ の増加率を比較する。

マスター定理を直感的に見てみよう

- 再帰による $aT(n/b)$ と $f(n)$ の増加率を比較する。
- 再帰による $aT(n/b)$ の漸近的増加率は $O(n^{\log_b a})$ である。

マスター定理を直感的に見てみよう

- 再帰による $aT(n/b)$ と $f(n)$ の増加率を比較する。
- 再帰による $aT(n/b)$ の漸近的増加率は $O(n^{\log_b a})$ である。
- $n^{\log_b a}$ を**分水嶺関数（watershed function）**という。

マスター定理を直感的に見てみよう

- 再帰による $aT(n/b)$ と $f(n)$ の増加率を比較する。
- 再帰による $aT(n/b)$ の漸近的増加率は $O(n^{\log_b a})$ である。
- $n^{\log_b a}$ を**分水嶺関数（watershed function）**という。
- 1の場合は $aT(n/b)$ の増加率が大きい。

マスター定理を直感的に見てみよう

- 再帰による $aT(n/b)$ と $f(n)$ の増加率を比較する。
- 再帰による $aT(n/b)$ の漸近的増加率は $O(n^{\log_b a})$ である。
- $n^{\log_b a}$ を**分水嶺関数（watershed function）**という。
- 1 の場合は $aT(n/b)$ の増加率が大きい。
- 2 の場合は同じぐらい。

マスター定理を直感的に見てみよう

- 再帰による $aT(n/b)$ と $f(n)$ の増加率を比較する。
- 再帰による $aT(n/b)$ の漸近的増加率は $O(n^{\log_b a})$ である。
- $n^{\log_b a}$ を**分水嶺関数（watershed function）**という。
- 1 の場合は $aT(n/b)$ の増加率が大きい。
- 2 の場合は同じぐらい。
- 3 の場合は $f(n)$ の増加率が大きい。

マスター定理を直感的に見てみよう

- 再帰による $aT(n/b)$ と $f(n)$ の増加率を比較する。
- 再帰による $aT(n/b)$ の漸近的増加率は $O(n^{\log_b a})$ である。
- $n^{\log_b a}$ を**分水嶺関数（watershed function）**という。
- 1 の場合は $aT(n/b)$ の増加率が大きい。
- 2 の場合は同じぐらい。
- 3 の場合は $f(n)$ の増加率が大きい。
- 1 と 3 の場合でいう増加率の差は**多項式的な差**以上なければならない。

マスター定理を使ってみる：マージソート

- 漸化式は $T(n) = 2T(n/2) + \Theta(n)$ である。

マスター定理を使ってみる：マージソート

- 漸化式は $T(n) = 2T(n/2) + \Theta(n)$ である。
- $a = 2$ と $b = 2$、よって分水嶺関数は $n^{\log_2 2} = n$。

マスター定理を使ってみる：マージソート

- 漸化式は $T(n) = 2T(n/2) + \Theta(n)$ である。
- $a = 2$ と $b = 2$、よって分水嶺関数は $n^{\log_2 2} = n$。
- $k = 0$ とおき $f(n) = \Theta(n) = \Theta(n^{\log_b a} \lg^k n)$、よって 2 の場合になる。

マスター定理を使ってみる：マージソート

- 漸化式は $T(n) = 2T(n/2) + \Theta(n)$ である。
- $a = 2$ と $b = 2$、よって分水嶺関数は $n^{\log_2 2} = n$。
- $k = 0$ とおき $f(n) = \Theta(n) = \Theta(n^{\log_b a} \lg^k n)$、よって 2 の場合になる。
- 従って、$T(n) = \Theta(n \lg n)$。

マスター定理を使ってみる：行列積の単純分割統治

- 漸化式は $T(n) = 8T(n/2) + \Theta(1)$ である。

マスター定理を使ってみる：行列積の単純分割統治

- 漸化式は $T(n) = 8T(n/2) + \Theta(1)$ である。
- $a = 8$ と $b = 2$、よって分水嶺関数は $n^{\log_2 8} = n^3$。

マスター定理を使ってみる：行列積の単純分割統治

- 漸化式は $T(n) = 8T(n/2) + \Theta(1)$ である。
- $a = 8$ と $b = 2$、よって分水嶺関数は $n^{\log_2 8} = n^3$。
- 任意の正の $\epsilon < 3$ をとれば $f(n) = O(n^{3-\epsilon})$ がなりたつ。

マスター定理を使ってみる：行列積の単純分割統治

- 漸化式は $T(n) = 8T(n/2) + \Theta(1)$ である。
- $a = 8$ と $b = 2$、よって分水嶺関数は $n^{\log_2 8} = n^3$。
- 任意の正の $\epsilon < 3$ をとれば $f(n) = O(n^{3-\epsilon})$ がなりたつ。
- 分水嶺関数 n^3 は $f(n) = \Theta(1)$ より、多項式的な増加率の差を持つ。よって 1 の場合になる。

マスター定理を使ってみる：行列積の単純分割統治

- 漸化式は $T(n) = 8T(n/2) + \Theta(1)$ である。
- $a = 8$ と $b = 2$、よって分水嶺関数は $n^{\log_2 8} = n^3$。
- 任意の正の $\epsilon < 3$ をとれば $f(n) = O(n^{3-\epsilon})$ がなりたつ。
- 分水嶺関数 n^3 は $f(n) = \Theta(1)$ より、多項式的な増加率の差を持つ。よって 1 の場合になる。
- 従って、$T(n) = \Theta(n^3)$。

マスター定理を使ってみる：行列積の Strassen のアルゴリズム

- 漸化式は $T(n) = 7T(n/2) + \Theta(n^2)$ である。

マスター定理を使ってみる：行列積の Strassen のアルゴリズム

- 漸化式は $T(n) = 7T(n/2) + \Theta(n^2)$ である。
- $a = 7$ と $b = 2$、よって分水嶺関数は $n^{\log_2 7} = n^{\lg 7}$。

マスター定理を使ってみる：行列積の Strassen のアルゴリズム

- 漸化式は $T(n) = 7T(n/2) + \Theta(n^2)$ である。
- $a = 7$ と $b = 2$、よって分水嶺関数は $n^{\log_2 7} = n^{\lg 7}$。
- $\lg 7 > 2$ なので、任意の正の $\epsilon < \lg 7 - 2$ をとれば $f(n) = O(n^{\lg 7 - \epsilon})$ がなりたつ。

マスター定理を使ってみる：行列積の Strassen のアルゴリズム

- 漸化式は $T(n) = 7T(n/2) + \Theta(n^2)$ である。
- $a = 7$ と $b = 2$、よって分水嶺関数は $n^{\log_2 7} = n^{\lg 7}$。
- $\lg 7 > 2$ なので、任意の正の $\epsilon < \lg 7 - 2$ をとれば $f(n) = O(n^{\lg 7 - \epsilon})$ がなりたつ。
- 分水嶺関数 $n^{\lg 7}$ は $f(n) = \Theta(n^2)$ より、多項式的な増加率の差を持つ。よって 1 の場合になる。

マスター定理を使ってみる：行列積の Strassen のアルゴリズム

- 漸化式は $T(n) = 7T(n/2) + \Theta(n^2)$ である。
- $a = 7$ と $b = 2$、よって分水嶺関数は $n^{\log_2 7} = n^{\lg 7}$。
- $\lg 7 > 2$ なので、任意の正の $\epsilon < \lg 7 - 2$ をとれば $f(n) = O(n^{\lg 7 - \epsilon})$ がなりたつ。
- 分水嶺関数 $n^{\lg 7}$ は $f(n) = \Theta(n^2)$ より、多項式的な増加率の差を持つ。よって 1 の場合になる。
- 従って、$T(n) = \Theta(n^{\lg 7})$。

飛ばす内容

- マスター定理の証明
- Akra-Bazzi 法