

ソフトウェア工学

2009年度 5セメスタ開講

1

履修にあたって

教科書:「アルゴリズムとデータ構造」

平田富夫著 森北出版

参考書:「データ構造とアルゴリズム」

エイホ他著、倍風館

講義種類:5セメスター開講、専門科目、

時間割:金曜3時限(12:50-14:20)

講義室:K325

担当:

草刈 良至

GI511(内線 2095)、kusakari@akita-pu.ac.jp

2

講義予定

1回 4/10(金)	2回 4/17(金)	3回 4/24(金)	4回 5/1(金)	5回 5/8(金)	6回 5/15(金)	7回 5/22(金)	8回 5/29(金)
	レポート 提出 (S0)		レポート 提出 (S1)			レポート 提出 (S2)	
6/5(金)	9回 6/12(金)	10回 6/19(金)	11回 6/26(金)	12回 7/3(金)	13回 7/10(金)	14回 7/17(金)	試験 7/31(金)
開学記 念日 (休講)		レポート 提出 (S3)			レポート 提出 (S4)		持込不 可

3

評価

- 授業内演習(出席)15%
- レポート25%
- 試験60%

4

本講義の目的

- よいソフトウェアを作成するための基礎を身に着ける。
- 良いソフトウェアであることの客観的な評価法を身に着ける。

5

本講義のレポート

- 主にC言語によるプログラミングが伴う。
- レポート作成の際には、プログラミング演習室を用いることができる。
 - ◆ただし、木曜日と、金曜日の午後は、3セメスターのプログラミング演習があるので、他の時間帯に利用すること。

6

1.アルゴリズム入門

7

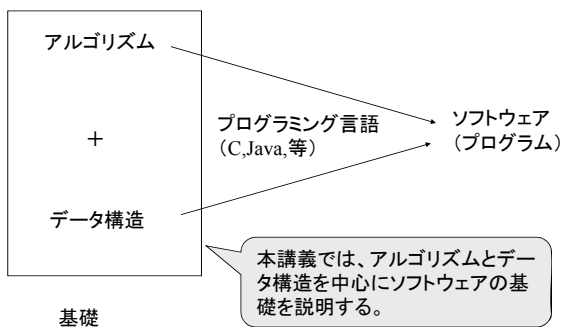
よいソフトウェアとは

本講義では、主にこの部分に注目する。

- 正しく動作するソフトウェア
- 同じ処理を速く実行できるソフトウェア(同じハードウェアで動作させた場合。)
- 同じ処理を少ないメモリで実行できるソフトウェア
- 再利用が可能なソフトウェア
- 誤動作のないソフトウェア
- 使いやすいソフトウェア
- 等々

8

ソフトウェア作成の基礎



本講義での主な注目点

- 正しく動作するプログラムの作成とアルゴリズムの正当性
- 高速に動作するプログラムの作成とアルゴリズムの評価
 - なお、アルゴリズムとは、計算機の基本操作の有限個の組み合わせである。すわち、機械的な手順で、有限であるもの。厳密には、チューリング機械やRAM(Random Access Machine)を用いて定義されるが、本講義では省略する。

10

アルゴリズムの解析

- 正当性
 - 数学的証明 帰納法や背理法
 - 実験的解析 実装とテスト
 - 速度の解析
 - 数学的解析 O記法による時間量解析
 - 実験的解析 実装と時間計測
- 講義で解説
- レポートで検証
- レポートで検証

11

アルゴリズムの計算量 (complexity)

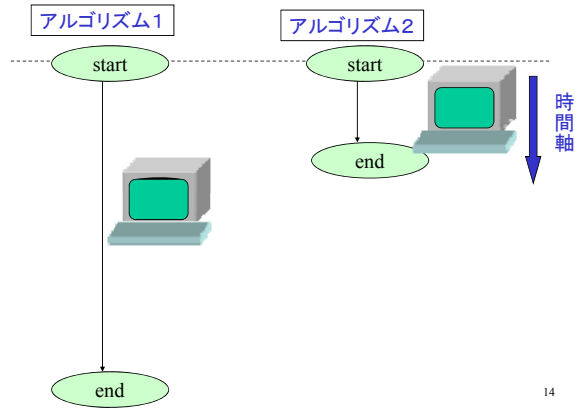
12

アルゴリズムの計算量1

- 時間計算量 (time complexity)
 - 総ステップ数 (基本演算の総数、アルゴリズムでは ∞ にはならない。)
 - 同じハードウェアでも速く実行できるプログラム作成のための指標。
- 領域計算量 (space complexity)
 - アルゴリズム実行時に、開始から終了までの間に使用するメモリやディスクなどの利用量
 - 記憶量ともいう。

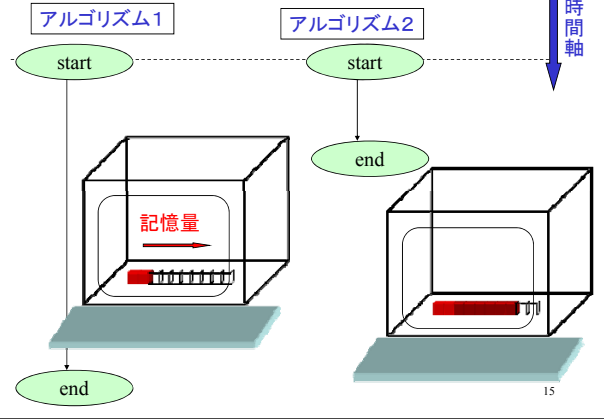
13

時間計算量



14

領域計算量



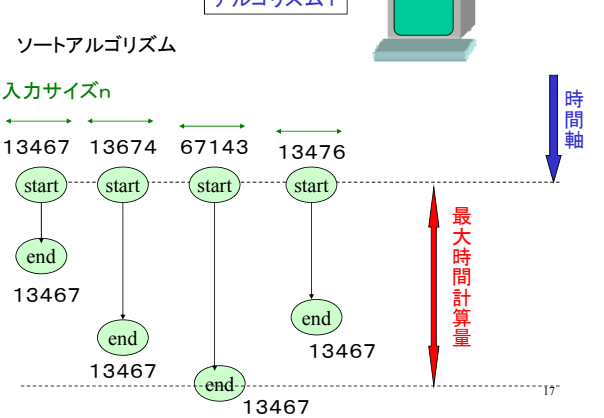
15

アルゴリズムの計算量2

- 最大時間計算量 (worst case time complexity)
 - 同じ入力サイズの問題に対して、最も遅く動作する場合を想定したときの時間計算量。
 - 最悪計算量ともいう。
- 平均時間計算量 (average case time complexity)
 - 同じ入力サイズの問題に対して、入力の分布を考えて、時間計算量を平均したもの。

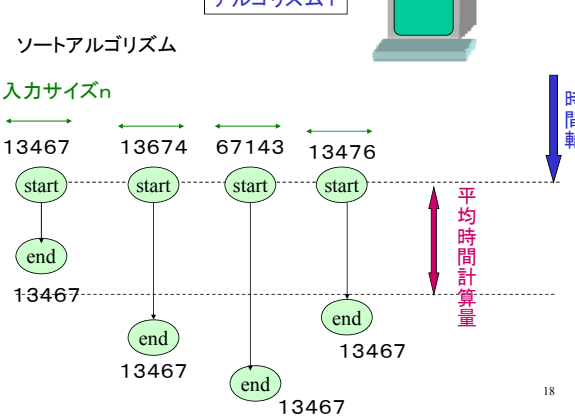
16

最大時間計算量



17

平均時間計算量



18

アルゴリズムの解析例

19

簡単なアルゴリズム例 (最大値を求める。)

アルゴリズムmax

```

1. big=A[0];
2. for(i=1;i<n;i++){
3.   if(A[i]>big){
4.     big=A[i];
5.   }
6. }
```

1回の代入

n回の比較

n-1回の比較

最悪n-1回の代入

最大時間計算量 $T(n) = 3n - 1$ のアルゴリズム

20

アルゴリズムmaxの正当性

次の命題を帰納法によって証明する。

命題1

forループがi回実行されたとき、
bigにはA[0]~A[i]の最大値が保持されている。

証明

基礎

i=0

このときは、bigにはA[0]が保持されており、明らかに命題は成り立つ。

帰納

i=kの時、命題1が成り立つと仮定する。(帰納法の仮定)
このとき、i=k+1を考える。

21

帰納法の仮定より、

$$\text{big} = \max\{A[0], A[1], \dots, A[k]\}$$

このとき、2つの場合に分けて考える。

場合1 A[k+1]>bigのとき。

このときは、アルゴリズムmaxの3. のif文の条件分岐が真なので、big=A[k+1]に更新される。
よって、k+1回目の繰り返し終了時には、
 $\text{big} = \max\{A[0], A[1], \dots, A[k+1]\}$

場合2 A[k+1]≤bigのとき。

$$\begin{aligned} & \max\{A[0], A[1], \dots, A[k+1]\} \\ &= \max\{\max\{A[0], A[1], \dots, A[k]\}, A[k+1]\} \\ &= \max\{\text{big}, A[k+1]\} \\ &= \text{big} \end{aligned}$$

どちらの場合も命題が成り立つ。

QED 22

アルゴリズムmaxの停止性

次の命題を証明する。

命題2

forループの反復部分は、丁度n-1回実行される。

証明

ループカウンタiは1からはじまる。
また、ループカウンタiが繰り返し事に1増加する。
ループカウンタがi=nになったときには、ループの反復部分は実行されない。したがって、丁度n-1回反復部分は実行される。

QED

命題1と命題2より、アルゴリズムmaxは正しいことがわかる。

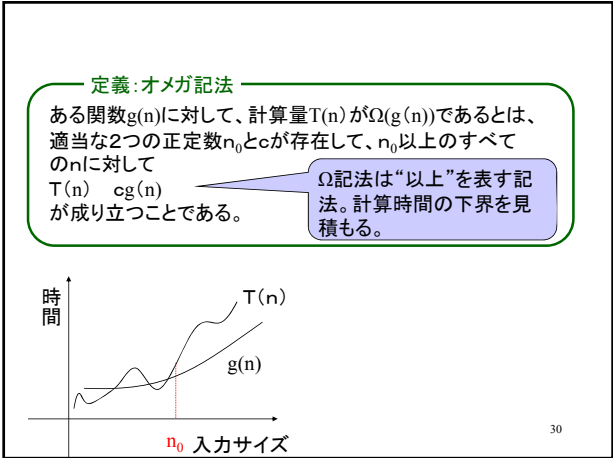
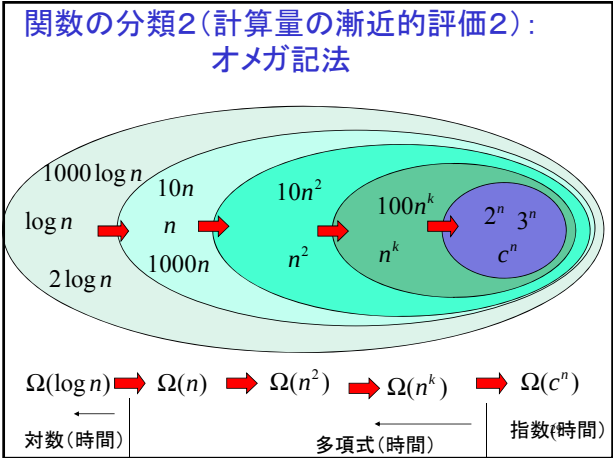
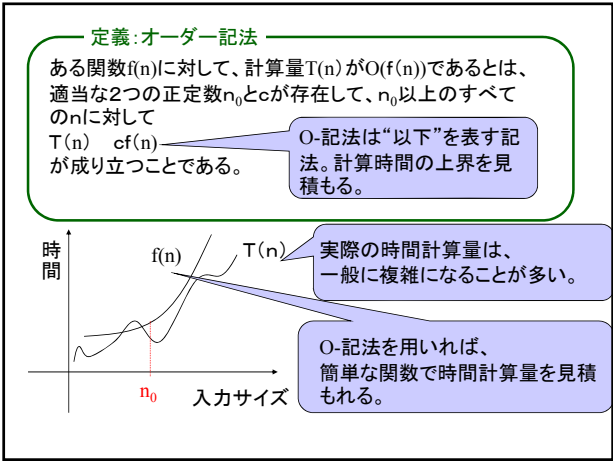
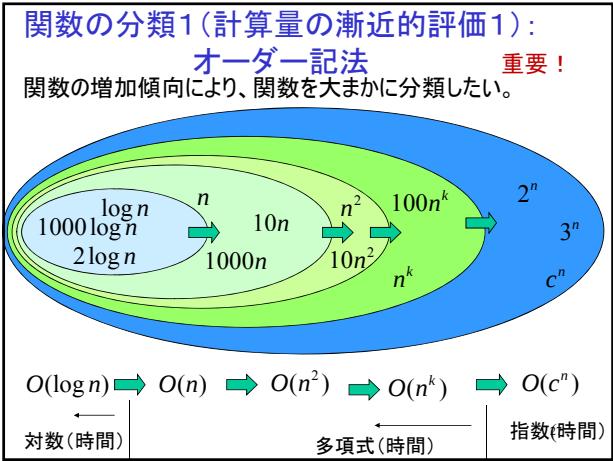
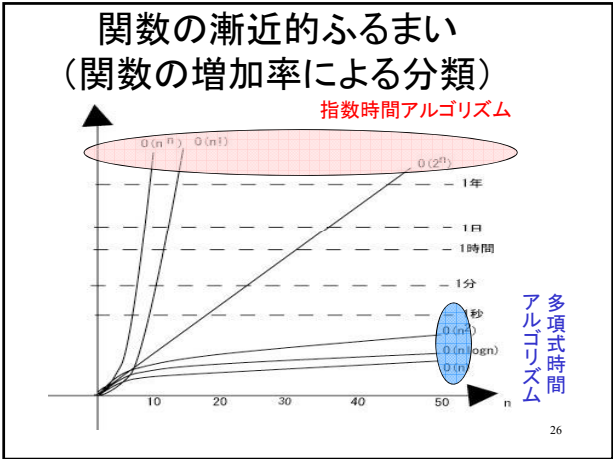
23

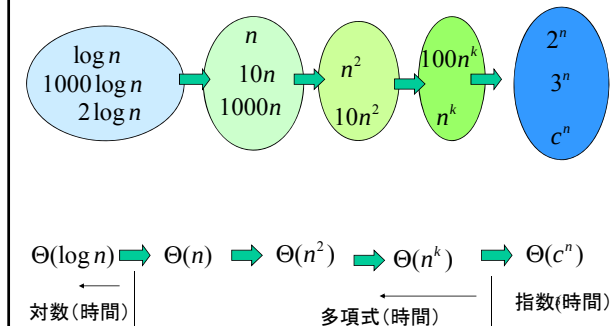
漸近的解析 (Asymptotic Analysis)

24

入力 サイズ	アルゴリ ズム	A n	B $n \log n$	C n^2	D n^3	E 2^n	F $n!$
$n = 10$		1.0×10^{-7}	3.3×10^{-7}	1.0×10^{-4}	1.0×10^{-5}	1.0×10^{-5}	0.04
15		1.5×10^{-7}	0.6×10^{-6}	2.3×10^{-7}	0.3×10^{-4}	3.3×10^{-4}	$3.6(h)$
20		2.0×10^{-7}	0.8×10^{-6}	0.4×10^{-5}	0.8×10^{-4}	1.1×10^{-2}	$772(y)$
30		3.0×10^{-7}	1.5×10^{-6}	0.9×10^{-5}	0.3×10^{-3}	11	
50		5.0×10^{-7}	2.8×10^{-6}	1.6×10^{-5}	1.3×10^{-3}	$4 \times 10^4(y)$	
100		1.0×10^{-6}	6.6×10^{-6}	1.0×10^{-4}	0.01		
500		5.0×10^{-6}	4.5×10^{-5}	2.5×10^{-3}	1.3		
1000		1.0×10^{-5}	0.1×10^{-3}	1.0×10^{-2}	10		
10000		1.0×10^{-4}	1.3×10^{-3}	1.0	$2.8(h)$		
10^5		1.0×10^{-3}	1.7×10^{-2}	100	$116(d)$		

100MIPSの計算機(1命令あたり 10^{-8} 秒) 単位: 秒(sec) 25

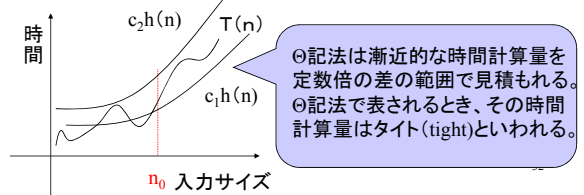


関数の分類3(計算量の漸近的評価3):
シータ記法

定義:シータ記法

ある関数 $h(n)$ に対して、計算量 $T(n)$ が $\Theta(h(n))$ であるとは、
適当な3つの正定数 n_0 , c_1 , c_2 が存在して、 n_0 以上のす
べての n に対して
 $c_1 h(n) \leq T(n) \leq c_2 h(n)$
が成り立つことである。

Θ記法は“ほぼ等し
い”を表す記法。



Θ記法は漸近的な時間計算量を
定数倍の差の範囲で見積もれる。
Θ記法で表されるとき、その時間
計算量はタイト(tight)といわれる。

O記法の例

(1) $f(n) = 2n$

$n_0 = 1, c = 2, g(n) = n$ とすれば、

$n \geq 1$ に対して、 $f(n) \leq 2g(n)$

よって、 $2n = O(n)$

(2) $f(n) = 5n^2 + 100n$

$n_0 = 100, c = 6, g(n) = n^2$ とすれば、

$n \geq 100$ に対して、 $f(n) \leq 6g(n)$

よって、 $5n^2 + 100n = O(n^2)$

注意2: 通常O記法では、最も簡単な関数で表す。

33

O記法の例

(3) $f(n) = 10000000000$

$n_0 = 1, c = 10000000000, g(n) = 1$ とすれば、

$n \geq 1$ に対して、 $f(n) \leq cg(n)$

よって、 $10000000000 = O(1)$

(4) $f(n) = 2^n + n^{100}$

$n_0 = 1000, c = 2, g(n) = 2^n$ とする。

$2^{1000} = (2^{10})^{100} = (1024)^{100} > (1000)^{100}$ なので、

$n \geq 1000$ に対して、 $f(n) \leq cg(n)$

よって、 $2^n + n^{100} = O(2^n)$

34

O記法の練習問題

次の数列の一般項(関数)をO記法で表
せ。

(1) $f(n) = 10n - 500$

(2) $f(n) = 10n\sqrt{n} + 100000n^2$

(3) $f(n) = 1000 \log_2 n + 10 \log_{10} n$

(4) $f(n) = \sqrt{n} + \log_2 n$

(5) $f(n) = 2^n + 3^n$

35

プログラムの漸近的評価

36

プログラムと漸近的評価

仮定1

プログラム内の加減算は、ある定数 C_1 時間以下で実行できる。

仮定2

プログラム内の乗除算は、ある定数 C_2 時間以下で実行できる。

仮定3

プログラム内の比較は、ある定数 C_3 時間以下で実行できる。

仮定4

プログラム内の代入は、ある定数 C_4 時間以下で実行できる。

プログラムでは、このように仮定できることが多い。

37

仮定1-4より、 $c > \max\{C_1, C_2, C_3, C_4\}$ なる C をとると、

プログラム内の4則演算、比較等はある定数時間以下で実行できる。

つよめて

プログラム内では、
繰り返し構造、
(再帰関数を含む)関数呼び出し、
以外は定数時間で実行できると仮定できることが多い。

38

プログラムにおける計算時間の漸近評価例

```
function1()
{
```

```
    for(k=0; k<n; k++)
```

```
    {
```

```
        ....
```

```
    }
```

```
}
```

function1の計算時間は、 $O(n)$ である。

forループは、この部分
だけで漸近時間計算
量が見積もれる。

この部分がn回
実行されることに
注意する。

39

プログラムにおける計算時間の漸近評価例2

```
function2()
{
```

```
    for(k=0; k<n; k++)
```

```
    {
```

```
        for(j=0; j<n; j++)
```

```
        {
```

```
            ....
```

```
        }
```

```
    }
```

```
}
```

function2の計算時間は、 $O(n^2)$ である。

この部分は n 回
実行されることに
注意する。

この部分は n^2 回
実行されることに
注意する。

この部分は n 回
実行されることに
注意する。

40

プログラムにおける計算時間の漸近評価例3

```
function3()
{
```

```
    for(k=0; k<n; k++)
```

```
    {
```

```
        for(j=0; j<k; j++)
```

```
        {
```

```
            .....
```

```
        }
```

```
    }
```

```
}
```

function3の計算時間 $f(n)$ を評価する。

$$f(n) = c(1 + 2 + 3 + \dots + n) = c \frac{n(n+1)}{2} = O(n^2)$$

41

外側のループカウンタ
が、内側のループ回
数に影響を与える。
一見、 n と無関係に
見える。

プログラムにおける計算時間の漸近評価例4

```
int function4(int n)
{
```

```
    if(n<1)
```

```
    {
```

```
        return(0);
```

```
    }
```

```
    else
```

```
    {
```

```
        function4(n-1);
```

```
    }
```

```
}
```

function4の計算時間 $f(n)$ を評価する。

$$\begin{cases} f(1) = c_1 \\ f(n) = f(n-1) + c_2 \end{cases}$$

この漸化式より、
 $f(n) = O(n)$ である。

再帰関数の時間計算量は、見た目では分かりにくい。

42

プログラムにおける計算時間の漸近評価例5

```
int function5(int n)
```

```
{
    if(n<1)
    {
        return(0);
    }
    else
    {
        function5(n/2);
    }
}
```

function5の計算時間 $f(n)$ を評価してみましょう。

$$\begin{cases} f(1) = c_1 \\ f(n) = f(\frac{n}{2}) + c_2 \end{cases}$$

この漸化式より、
 $f(n) = O(\log n)$ である。

43

プログラムにおける計算時間の漸近評価例6

```
int function6(int n)
```

```
{
    if(n<1)
    {
        return;
    }
    else
    {
        function6(n-1);
        function6(n-1);
    }
}
```

function6の計算時間 $f(n)$ を評価してみましょう。

$$\begin{cases} f(1) = c_1 \\ f(n) = f(n-1) + f(n-1) + c_2 \\ \quad = 2f(n-1) + c_2 \end{cases}$$

この漸化式より、
 $f(n) = O(2^n)$ である。

44

プログラムにおける計算時間の漸近評価練習1

次のプログラムの計算時間を O 記法で求めよ。
ただし、入力サイズは仮引数 n に入っている数とする。

45

(1)

```
exercise1(int n)
{
    for(j=0; j<n; j++)
    {
        for(k=0; k<n; k++)
        {
            .....
        }
    }
    for(l=0; l<n; l++)
    {
        x x x x
    }
}
```

46

(2)

```
exercise2(int n)
{
    if(n<2)
    {
        return;
    }
    else
    {
        exercise2(n-1);
        exercise2(n-2);
    }
}
```

47

アルゴリズムの入力について

- ・問題と問題例
- ・入力サイズ

48

問題と問題例

(problem and problem instances)

問題: 現実の問題を定義したもの。
同じような入力と出力の関係を定めたもの。

- 与えられた並んだ数値を順番にならべる。
→ ソート問題 (入力: 与えられた数値の列、出力: 順序列)
- 2つの数字の最大公約数を求める。
→ gcd問題 (入力: 2つの整数、出力: 1つの最大公約数)
- 数の集合から最大値を求める。
→ 最大値問題 (入力: 数の集合、出力: 入力中の最大値)

...

49

問題例: 具体的に数値を与えたもの。
問題は、問題例の集合としてとらえられる。

ソート問題

問題

問題例

3 4 2 8 7 → 2 3 4 7 8
1 2 9 7 3 5 6 → 1 2 3 5 6 7 9
4 2 → 2 4
7 1 3 8 → 1 3 7 8

50

最大値問題

問題

問題例

3 4 2 8 7 → 8
1 2 9 7 3 5 6 → 9
4 2 → 4
7 1 3 8 → 8

51

入力サイズ

入力をコンピュータで表現するときの大きさ。
一つ問題例を定めると入力サイズも定まる。

ソート問題

入力サイズ

3 4 2 8 7 (5)
7 1 3 8 (4)
4 2 (2)
1 2 9 7 3 5 6 (7)

52

本講義では主にこの基準を用いる

- 一様コスト基準 (一様コストモデル)
どの数の計算も一定時間 (定数時間) できるとき
(一つの数の入力サイズは1)
- 対数コスト基準 (対数コストモデル)
数の表現を桁数まで考えて数を扱う。
桁の大きい数同士の計算は大変なので。
(数aの入力サイズはlog a)

この基準を用いるときは、その都度こたわる

53

対数コストモデルについて
(計算機内での数の表現と桁数)

10進数 $d_n d_{n-1} \dots d_0$ $d_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

2進数 $b_m b_{m-1} \dots b_0$ $b_i \in \{0, 1\}$

互換

上のように相互変換されるとき、
 $m = O(n)$
である。

n : 10進数での桁数
 m : 2進数での桁数

54

証明

$$A \equiv d_n \times 10^n + d_{n-1} \times 10^{n-1} + \dots + d_0 = b_m \times 2^m + \dots + b_0$$

$2^m \leq A < 2^{m+1}$ に注意して、底2の対数をとる。

$$\log_2 2^m \leq \log_2 A < \log_2 2^{m+1}$$

$$\therefore m \leq \log_2 A < m+1$$

また、 $10^n \leq A < 10^{n+1}$ に注意して、底2の対数をとる。

$$\log_2 10^n \leq \log_2 A < \log_2 10^{n+1}$$

$$\therefore n \log_2 10 \leq \log_2 A < (n+1) \log_2 10$$

$c \equiv \log_2 10$ とおく。

$$m \leq \log_2 A < (n+1) \log_2 10 = c(n+1)$$

$$\therefore m < c(n+1)$$

55
QED

・最大値問題の入力サイズ

33 424 21 996 1242 → 1242

一様コスト基準
5本講義では、
主にこちらを用いる。

33 424 21 996 1242

↔ ↔ ↔ ↔ ↔

2

3

2

3

4

↔ ↔ ↔ ↔ ↔

$$2+3+2+3+4$$

$$=14$$

対数コスト基準

56