情報数理概説資料 1

松本　眞 広島大学理学部数学科 m-mat アット math.sci.hiroshima-u.ac.jp
このノートと、講義に関するプログラムが
http://www.math.sci.hiroshima-u.ac.jp/~m-mat/TEACH/teach.html
からダウンロードできる。
参考書として [1]（このノートのラスト参照）を用いる。

1　万物は数である

（参考書１．１節に対応。１．２、１．３は省略する。）
「万物は数である」ピタゴラス、BC580–500.
計算機＝コンピュータでは、全てを２進数であらわす。
１６進数 0123456789abcdef
次のようなテキストファイル

```
01234567
89
abcdefgh
ABCDEFGH
あいうえお
アイウエオ
漢字
```

が、実際には次のような「数の列」として記述されている。シフト JIS の場合

```
30 31 32 33 34 35 36 37 0d 0a 38 39 0d 0a 61 62
63 64 65 66 67 68 0d 0a 41 42 43 44 45 46 47 48
0d 0a 82 a0 82 a2 82 a4 82 a6 82 a8 0d 0a 83 41
83 43 83 45 83 47 83 49 0d 0a 8a bf 8e 9a 0d 0a
```

ISO-2022-JP (JIS) の場合

```
30 31 32 33 34 35 36 37 0d 0a 38 39 0d 0a 61 62
63 64 65 66 67 68 0d 0a 41 42 43 44 45 46 47 48
0d 0a 1b 24 42 24 22 24 24 24 26 24 28 24 2a 1b
28 42 0d 0a 1b 24 42 25 22 25 24 25 26 25 28 25
2a 1b 28 42 0d 0a 1b 24 42 34 41 3b 7a 1b 28 42
0d 0a
```

解説しないが、ひらがなも漢字も、うまく工夫して 2 進数で表している。

2　プログラム

2.1　最大公約数（参考書２．１節）

プログラム 2.1. gcd2-1a.c
　最大公約数を求める全数チェック式アルゴリズム

```
#include <stdio.h>
#include <stdlib.h>

int gcd(int a, int b) {
    int i;
    int n, ans;

    if (a < b) {
        n = a;
    } else {
        n = b;
    }
    for (i = 1; i <= n; i++) {
        if (a % i == 0 && b % i == 0) {
            ans = i;
        }
    }
    return ans;
}

int main(int argc, char *argv[]) {
    int a, b;

    if (argc <= 2) {
        printf("引数が足りません\n");
        return 1;
    };
    a = strtol(argv[1], NULL, 10);
    b = strtol(argv[2], NULL, 10);
    printf("gcd(%d, %d) = %d\n", a, b, gcd(a, b));
    return 0;
}
```

プログラム 2.2. gcd2-1b.c
最大公約数を求める再帰呼び出しによるプログラム

```
#include <stdio.h>
#include <stdlib.h>

int gcd(int a, int b) {
    if (b == 0) {
        return a;
    } else {
        return gcd(b, a % b);
    }
}

int main(int argc, char *argv[]) {
    int a, b;

    if (argc <= 2) {
        printf("引数が足りません\n");
        return 1;
    };
    a = strtol(argv[1], NULL, 10);
    b = strtol(argv[2], NULL, 10);
    printf("gcd(%d, %d) = %d\n", a, b, gcd(a, b));
    return 0;
}
```

プログラム 2.3. gcd2-1c.c
最大公約数を求める while 文によるプログラム

```
#include <stdio.h>
#include <stdlib.h>

int gcd(int x, int y) {
    int r;

    r = x % y;
    while (r > 0) {
        x = y;
        y = r;
        r = x % y;
    }
    return y;
}

int main(int argc, char *argv[]) {
    int a, b;

    if (argc <= 2) {
        printf("引数が足りません\n");
        return 1;
    };
    a = strtol(argv[1], NULL, 10);
    b = strtol(argv[2], NULL, 10);
    printf("gcd(%d, %d) = %d\n", a, b, gcd(a, b));
    return 0;
}
```

3　コンパイル・実行

上の gcd に関連するプログラムが、http://www.math.sci.hiroshima-u.ac.jp/~m-mat/TEACH のホムペに行って、一番下の方の 2007 年度後期「情報数理概説」参考プログラムの「参考プログラム」からダウンロードできる。Web ブラウザを使って gcd2-1a.c, gcd2-1b.c, gcd2-1c.c の三つの C 言語プログラムをダウンロードしよう。設定によるが、デスクトップにダウンロードされると思う。

プログラムがちらからないよう、一つこの授業用のディレクトリを作り、ダウンロード後にはそのディレクトリ（フォルダ）にしまうとよい。

フォルダの生成には左上の「ホーム」をダブルクリックし、「ファイル」から「フォルダの生成」をおこない、好きな名前 (英字が無難) をいれるとよい。

プログラムを実行するには、「アプリケーション」から「システムツール」から「GNOME 端末」をえらぶ。

```
[]$
```

にコマンドを入力する。

```
[]$ cd <いま作ったディレクトリ名>
```

でそのディレクトリに移動する。そこで

```
[]$ gcc gcd2-1a.c -o gcd-a
```

と入力する。これは、“gcd2-1a.c” というファイルをコンパイルし、gcd-a という名の実行可能ファイルを生成せよ、という命令である。

```
[]$ ./gcd-a 120 78
```

と入力し Enter(改行) すると、

```
gcd(120,78)=6
```

といった答えを得る。

では、

```
[]$ ./gcd-a 120000000 78000000
```

の答えはなにか？さらに１０倍して

```
[]$ ./gcd-a 1200000000 780000000
```

の答えを求めてみよう。
　時間計測コマンド time を使うと、

```
[]$ time <コマンド名>
```

と入力することで、プログラムの実行時間を知ることができる。たとえば、

```
[]$ time ./gcd-a 120000000 78000000
```

と入力してみよう。

```
real 0m1.077s
user 0m0.270s
sys 0m0.770s
```

といった具合に実行時間が出力される。「０分と 1.077 秒かかりました」と読めばよい。
　入力をそれぞれ１０倍するとどのくらい時間がかかるか？
　gcd2-1b.c や gcd2-1c.c もコンパイルして、実行してみよう。動作がわかるように、途中経過を出力する printf 文が入っているので、どう動作しているかわかるだろう。

課題 3.1. 以下について、実行・考察せよ。

- 動作時間を比べる。
- gcd2-1b.c を使って、さまざまな入力に対する動作を観察する。
- となりあった二つの Fibonacci 数 (参考書４９ページ参照) を入力し、「なかなか小さくならない」ことを確認する。
- $2^n - 1$ の形の数 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047, 4095, 8191, 16383, 32767, 65535, ... の最大公約数を求めてみることで、なんらかの法則性を発見する。

4　計算量

プログラム 2.1 とプログラム 2.2 のスピードはどのくらい違うか。

→計算量、計算量のオーダーの概念が必要

入力 n, m $(n \geq m)$ に対し、プログラム 2.1 では m 回のループを実行する。一回のループで 2 回の剰余計算を行うから、$2m$ 回の剰余計算を行う。

同じ入力に対し、プログラム 2.1 では、何回の剰余計算を行うか簡単な式ではかけない。しかし、参考書４８ページ定理 2.2 のように、たかだか（多くても、の意味）$2\log m + 2$ 回の剰余計算しか使わないことがわかる。（ここで、数学にはあるまじきことだが、このノートや参考書では log は $\log_2$ の意味である。）

課題 4.1. じつは、より小さく、$\log m / \log(\frac{1+\sqrt{5}}{2})^{-1}$ 程度の回数の剰余計算しかいらないことを示せ。$(1.44042 \log m)$ くらい。

このように、ある計算を基本単位として（上の例では剰余計算を使った）、計算に必要なステップ数を測ることを「時間的計算量」(time computational complexity) という。剰余算のほかに、足し算も数えたり、いろいろなバリエーションがある。入力によって計算量は変わるわけだが、「入力のサイズに対して定まる計算量」は関数である。

入力のサイズの単位：その入力を 2 進表現するするのに必要な bit 数であらわす。(理由：一般のプログラムでは，入力は数とは限らない、文字データかもしれないから。　そして、二つの入力があったときに、その合計サイズは「和」であるべき　だから。)

例：入力 m, n に対し、そのサイズは $\log m + \log n$ で与えられる。

（以上の「サイズ」の定義はなにかあいまいだが、気にしない。厳密にするには、「入力」の定義を「2 進列データ」とするべきである。そして、エンコードの方法＝数を 2 進列に変換する方法、を一つ特定する必要がある。）

時間計算量とは、入力のサイズに対し、そのアルゴリズムで（注目すべき基本計算の集合を決めて、それらに関して数えて）何回の計算が行われるかをあらわす関数である。

例 4.2. `gcd2-1a.c` の入力の小さいほうにのみ着目するとき、入力サイズ M ならば $(m = 2^M)$、時間計算量は $m = 2^M$ である。このように、入力サイズ M に対して、時間計算量が M の指数関数 $(a^M, a > 1$ になるとき、「指数時間アルゴリズム」という。

`gcd2-1b.c` の入力の小さいほうにのみ着目するとき、入力サイズ M ならば $(m = 2^M)$、時間計算量は高々$2M + 2$ である（参考書 P48 定理 2.2)。一般に、入力サイズの一次式によって時間計算量が上から抑えられるとき、「線形時間

アルゴリズム」という。入力サイズの多項式によって時間計算量が上から抑えられるとき、「多項式時間アルゴリズム」という。

数学や、物理学でいう「ランダウの O（オーダー）記号」を使うと、慣れると便利である。

定義 4.3. $f(n)$ を、自然数を定義域とし（有限個を除いて）正の実数に値を取る関数とする。$O(f(n))$ で、同様の性質を持つ関数の集合で、次のように定義されるものをあらわす。

$g(n) \in O(f(n)) \Leftrightarrow \exists C$ により $(g(n) < Cf(n)$ が十分大きな n で常に成立)

右辺の条件は、もっと厳密に書けば

$$\exists C \in \mathbb{R} \exists n_0 \in \mathbb{N}(n > n_0 \Rightarrow g(n) < Cf(n))$$

である。

例 4.4.

$$\lim_{n\to\infty} g(n)/f(n) < C$$

ならば、$g(n) \in O(f(n))$.

$O(1)$ は、自然数上有界な関数の集合。

`gcd2-1a.c` は指数時間アルゴリズム。`gcd2-1b.c` は線形時間アルゴリズム。

命題 4.5.

- $O(C) = O(1)$
- $O(f(n)) \times O(g(n)) \subset O(f(n)g(n))$
- $O(f(n)) \subset O(g(n))$ となる必要十分条件は、$f(n) \in O(g(n))$。

問題 4.6. $O(f(n)) = O(g(n))$ となる必要十分条件を、epsilon-delta 論法で記述せよ。

$g(n)/f(n)$ が $n \to \infty$ のとき 0 でない有限の値に収束するなら、上の条件が満たされることを示せ。

参考文献

[1] 渡辺治著　「教養としてのコンピュータサイエンス」（サイエンス社）