データ構造とアルゴリズム
(第2回)

http://coconut.sys.eng.shizuoka.ac.jp/algo/06/

静岡大学工学部

安藤和敏

2006.12.04

第1章アルゴリズムの基礎

1.1 アルゴリズムとは
1.2 アルゴリズムの評価基準
1.3 計算量の漸近的評価
1.4 アルゴリズムの記述

第1章アルゴリズムの基礎

1.1 アルゴリズムとは
1.2 アルゴリズムの評価基準
1.3 計算量の漸近的評価
1.4 アルゴリズムの記述

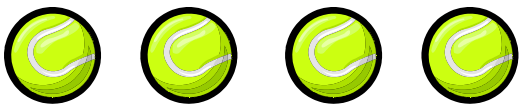
アルゴリズムの評価基準

問題1.1からもわかるように，一般に一つの問題を解くアルゴリズムは複数存在する．そうした場合に，どのアルゴリズムを選択すれば良いか？この，アルゴリズムの良し悪しを決める基準を**アルゴリズムの評価基準**と呼ぶ．

アルゴリズムの評価基準のひとつに，“計算時間の短さ”がある．これは直感的にも納得できるであろう．

問題1.2

*n*個のテニスボールがある．このテニスボール1個の重さは100gであるが，*n*個のうち1つだけ重さが95gの不良品である．重さが測定できるはかりを用いて，この不良品のテニスボールをみつけよ．



アルゴリズム1.2

入力: n**個のテニスボール** $\{b_1, b_2, \ldots, b_n\}$

アルゴリズム:

(1) i=1**とする.**

(2) **テニスボール**b_i **をはかりに載せる.**

(3) b_i **の重さが**100g**ならば,** i**を**1**だけ増加させて,** (2),(3)**の操作を繰り返す.** b_i **の重さが**95g**ならば, そのボールを不良品としてアルゴリズムを終了する.**

アルゴリズム1.3

入力: n**個のテニスボール** $\{b_1, b_2, \ldots, b_n\}$

アルゴリズム:

(1) **テニスボールを約半分の2つの集合**

$B_1 = \{b_1, b_2, \cdots, b_{\lceil \frac{n}{2} \rceil}\}, B_2 = \{b_{\lceil \frac{n}{2} \rceil + 1}, b_{\lceil \frac{n}{2} \rceil + 2}, \cdots, b_n\}$

に分ける.

(2) **テニスボールの集合** B_1 **をはかりに載せる.**

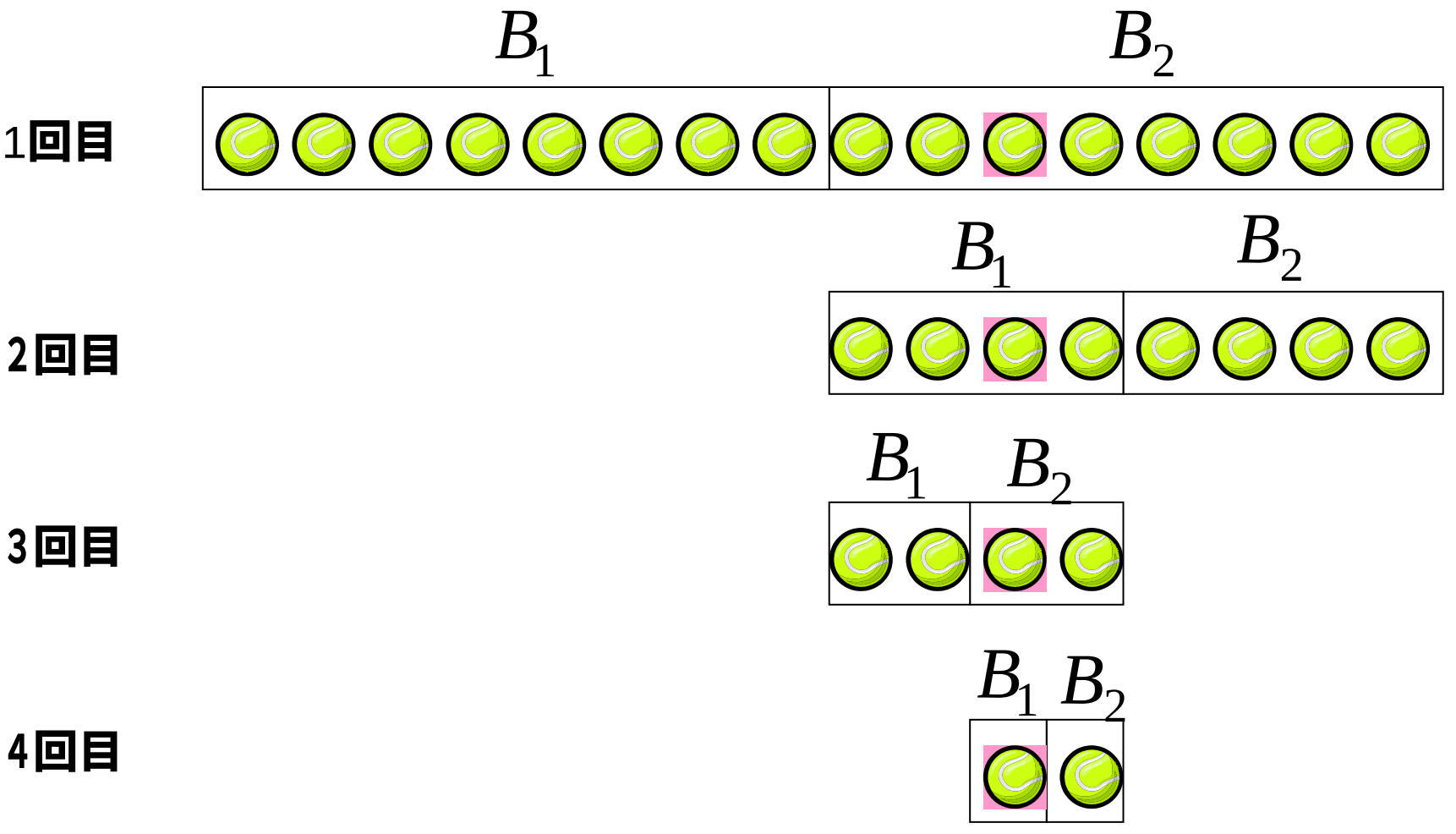
(3) B_1 **の重さが**100**の倍数ならば,** B_2 **の中に不良品があり,** B_1 **の重さが**100**の倍数でなければ, 不良品は**B_1 **の中にある. このとき, 不良品の含まれているほうのボールの集合に対して, 以下の操作を行う.**

アルゴリズム1.3 (続き)

(a) **不良品の含まれているボールの集合に一つのボールしかなければ，そのボールを不良品としてアルゴリズムを終了する．**

(b) **不良品の含まれているボールの集合に複数のボールが含まれていれば，そのボールの集合を** (1) **と同様に2つの集合** B_1 **と** B_2 **に分けて，** (2) **，** (3) **の操作を繰り返す．**

アルゴリズム1.3の動き



(2) の繰返しの回数

簡単のために n は2のべき乗であると仮定する.

(2) で, はかりに載せるテニスボールの数は, 繰返しごとに半減していく. したがって, k 回目の(2),(3)を実行したときに, 不良品の可能性のあるテニスボールの個数は,

$$\left(\frac{1}{2}\right)^k \times n$$

である. これが1に等しくなったときにアルゴリズムは終了する.

(2) の繰返しの回数 (続き)

したがって，アルゴリズムが終了するまでの繰返し回数 k は

$$\left(\frac{1}{2}\right)^k \times n = 1$$

を満たす．

上式を書き換えると，$n = 2^k$ であるから，

$$\log_2 n = k$$

を得る．

アルゴリズムの実行時間

“はかりに載せて重さを量る”という操作が10秒で実行できると仮定し，そのほかの操作の時間は無視できると仮定する．

テニスボールの数 n	アルゴリズム1.2		アルゴリズム1.3
	最良	最悪	
10	10秒	100秒	10×log10　40秒
100	10秒	1000秒	10×log100　70秒
1000	10秒	10000秒	10×log1000　100秒
10000	10秒	100000秒	10×log10000　140秒
100000	10秒	1000000秒	10×log100000　170秒

アルゴリズムの時間計算量

このように，アルゴリズムの実行時間は，アルゴリズムに対する入力サイズ (先ほどのアルゴリズムでは n) に依存する．n の関数で表現された実行時間を，そのアルゴリズムの**時間計算量**と呼ぶ．

	アルゴリズム1.2	アルゴリズム1.3
最良時間計算量	10	$10\log_2 n$
最悪時間計算量	$10n$	$10\log_2 n$

時間計算量は，アルゴリズム1.2のように，同じサイズ n の入力でも異なる場合がある．その場合，最も速くそのアルゴリズムを実行できる入力の時間計算量を**最良時間計算量**と呼んで，最も時間のかかる入力に対する時間計算量を，**最悪時間計算量**と呼ぶ．以降では，特に断らない限り，時間計算量は，最悪時間計算量を指すものとする．

第1章アルゴリズムの基礎

1.1 アルゴリズムとは
1.2 アルゴリズムの評価基準
1.3 計算量の漸近的評価
1.4 アルゴリズムの記述

時間計算量を求めることが難しい

一般に，あるアルゴリズムの正確な時間計算量を求めることは，非常に困難である．

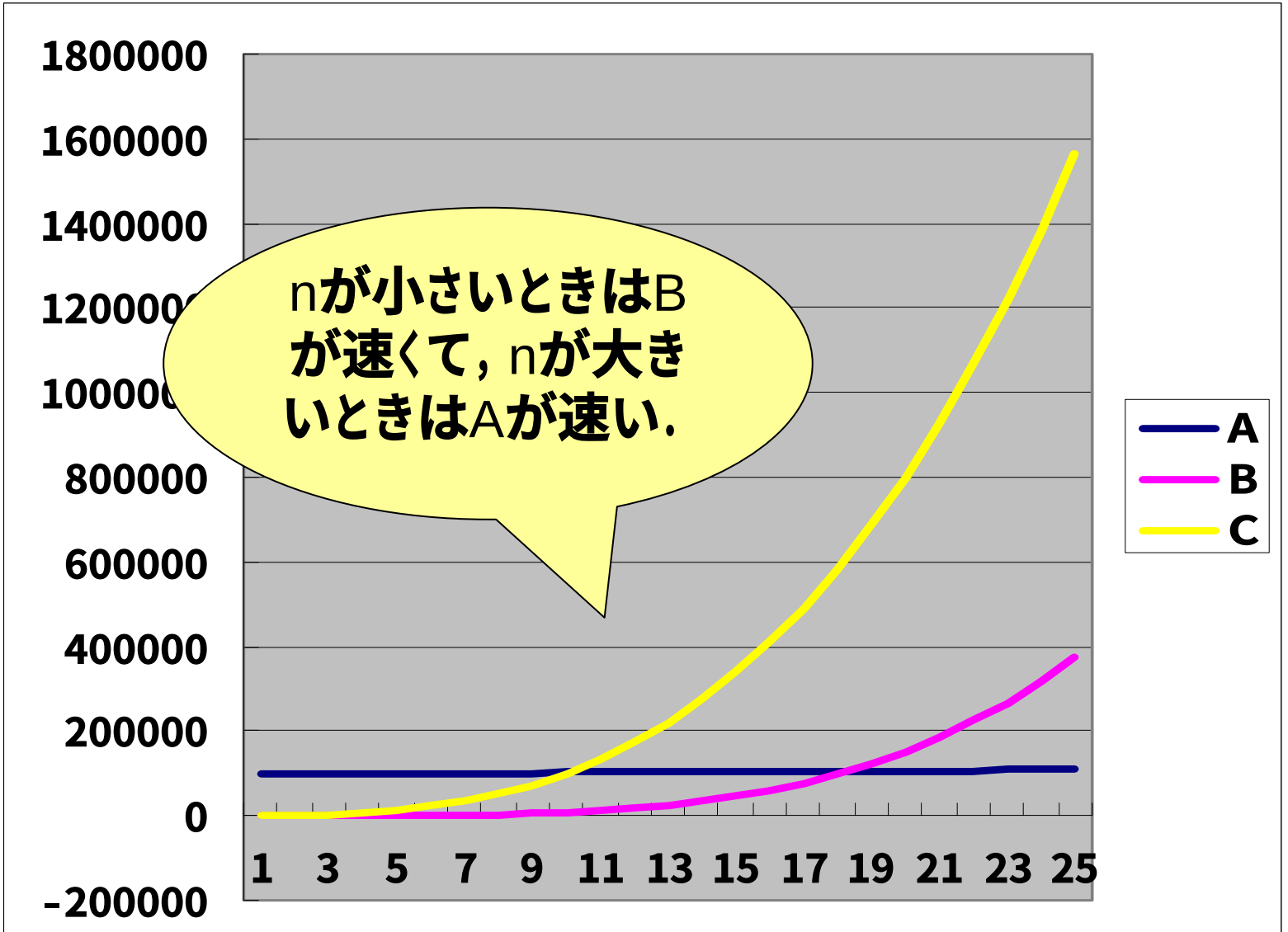
また，正確な時間計算量が求められたとしても，一般に時間計算量どうしの比較は困難である．

時間計算量を比較するのも難しい

ある問題に対して，A，B，Cの3つのアルゴリズムが存在して，以下のような時間計算量を持っているとする．

アルゴリズムA：$10n^2+100n+10000$
アルゴリズムB：n^4-n^3-n
アルゴリズムC：$100n^3$

3つのアルゴリズムの比較



漸近的な時間計算量

問題のサイズが小さいときは，どのアルゴリズムを用いても，あまり大差はない．

むしろ，nが非常に大きい場合（nを無限大に近づけたときの極限）で比べる．

このように，nが無限大に近いところでの，時間計算量を**漸近的な時間計算量**と呼ぶ．

漸近的な時間計算量は，次に定義される**オーダ記法**を用いて表現される．

オーダ記法

入力サイズ n の関数として表される時間計算量 $T(n)$ が，ある関数 $f(n)$ に対して $O(f(n))$ であるとは，適当な2つの正の定数n_0 と c が存在して，n_0 以上の全ての n に対して $T(n) \leqq cf(n)$ が成り立つことである．

例：アルゴリズムCの時間計算量は，
$T(n) = 100n^3$**であるから，アルゴリズムCは**$O(n^3)$**．**

例：アルゴリズムBの時間計算量は，
$T(n) = n^4-n^3-n$**であるから，アルゴリズムBは**$O(n^4)$**．**

例：アルゴリズムAの時間計算量は，
$T(n) = 10n^2+100n+10000$**であるから，アルゴリズムAは**$O(n^2)$**．**

オーダ記法の簡単な理解

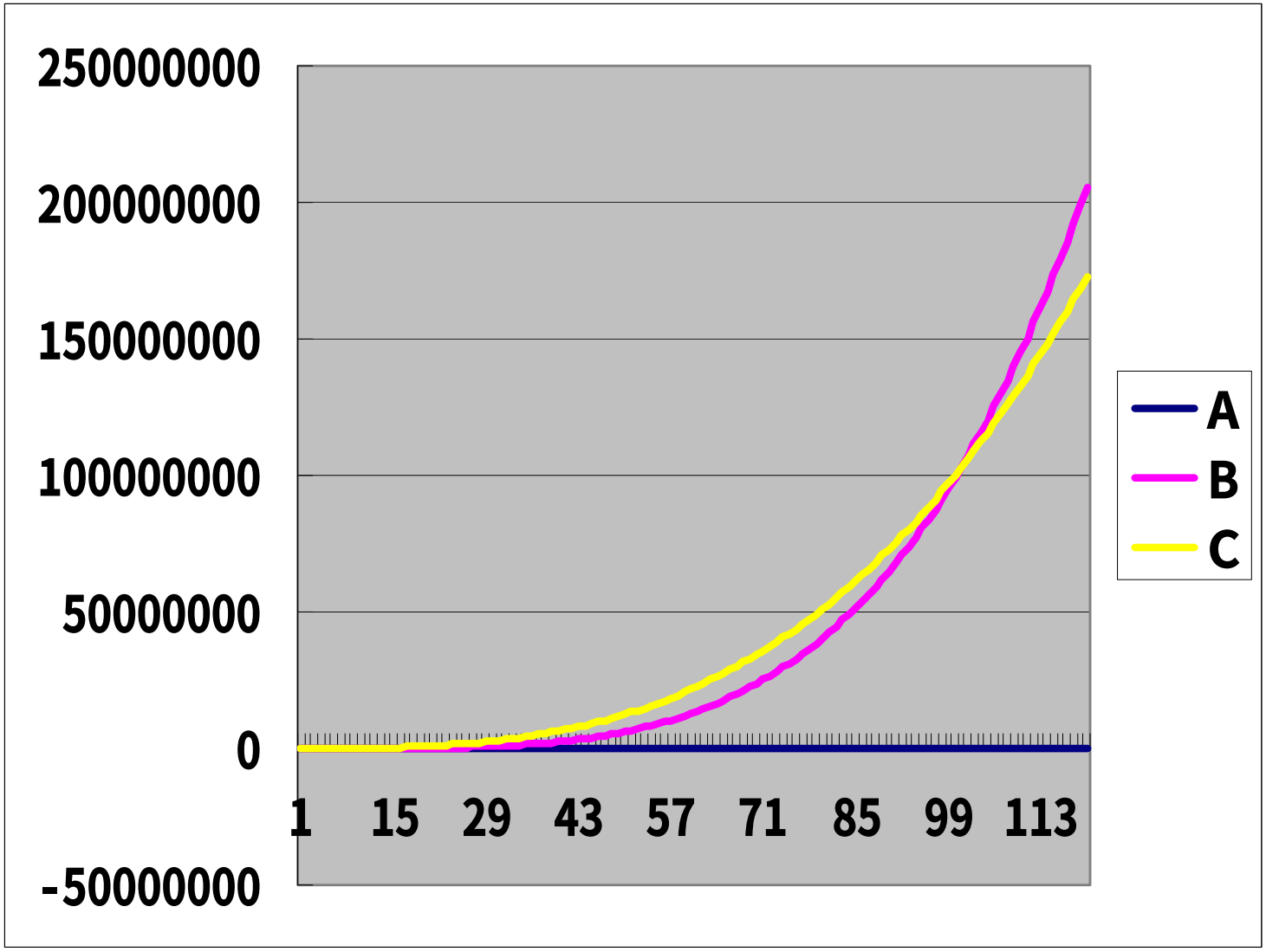
アルゴリズムの時間計算量が $T(n)$ **であったとしよう．**
$T(n)$ **の中で主要項（**n **を無限大に飛ばしたときに最も大きな項）を見つける．**
この主要項の係数を削除した関数が $f(n)$ **であるとき，このアルゴリズムの時間計算量は** $O(f(n))$ **であるという．**

例：アルゴリズムCの時間計算量は，
$T(n) = 100n^3$**であるから，アルゴリズムCは**$O(n^3)$**．**

例：アルゴリズムBの時間計算量は，
$T(n) = n^4-n^3-n$**であるから，アルゴリズムBは**$O(n^4)$**．**

例：アルゴリズムAの時間計算量は，
$T(n) = 10n^2+100n+10000$**であるから，アルゴリズムAは**$O(n^2)$**．**

アルゴリズムA,B,C


250000000
200000000
150000000
100000000
50000000
0
-50000000
1
15
29
43
57
71
85
99
113
A
B
C

代表的な関数の漸近的な大小関係

$$\log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \cdots < 2^n < n!$$

$f(n)$ がn に依存しない定数である場合は$O(1)$と記述して，その時間計算量を定数時間と呼ぶ．

アルゴリズム1.2と1.3の時間計算量

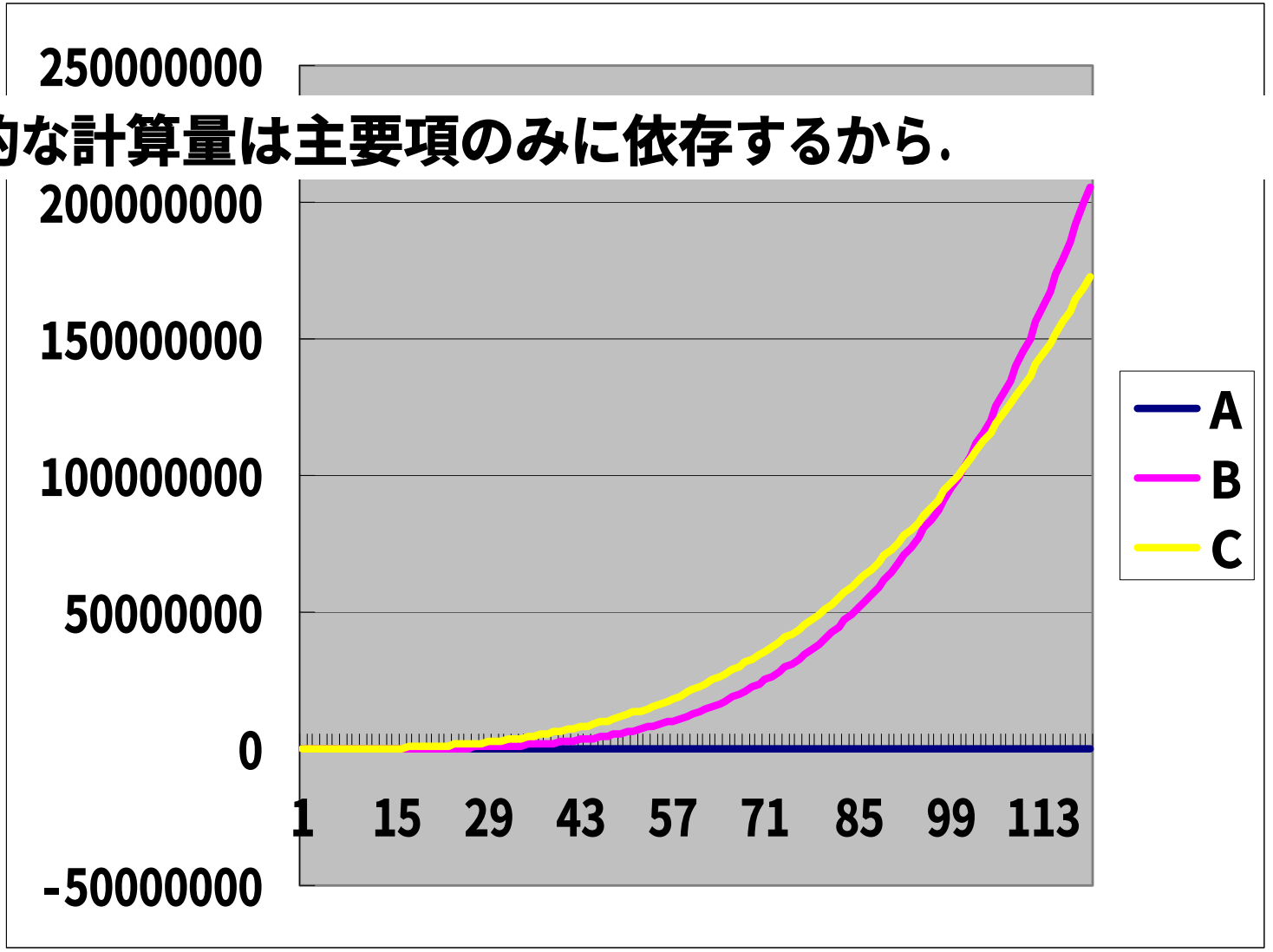
	アルゴリズム1.2	アルゴリズム1.3
最良時間計算量	10	$10\log_2 n$
最悪時間計算量	$10n$	$10\log_2 n$

	アルゴリズム1.2	アルゴリズム1.3
最良時間計算量	$O(1)$	$O(\log n)$
最悪時間計算量	$O(n)$	$O(\log n)$

注意: $\log_2 n = \dfrac{\log n}{\log 2}$ だから，$\log_2 n = O(\log n)$.

なぜオーダ記法なのか

漸近的な計算量は主要項のみに依存するから.



第1章アルゴリズムの基礎

1.1 アルゴリズムとは
1.2 アルゴリズムの評価基準
1.3 計算量の漸近的評価
1.4 アルゴリズムの記述

アルゴリズムの記述

以降では，アルゴリズムを記述するために，C言語に似た”擬似言語”を使用する．

本当のプログラミング言語ではないので，もちろん計算機で実行できない．しかし例えば，C言語に書き直すことは容易である　(僕にとっては)．

基本演算

算術演算（加減乗除など）や論理演算（AND や OR），及び，入出力命令は，O(1)時間で実行できると仮定する．

代入

変数への値の代入は “=“ という記号で表す. 二つの値が等しいという条件は ”==“ で, 二つの値が異なるという条件は “!=“ で表す.

例1 : 変数 x **に** 10 **を加算して代入する.**

```
x = x + 10;
```

例2 : 変数 x **が** 10 **と等しいという条件.**

```
x == 10
```

for文

for (**変数の初期値設定; 繰り返し継続条件; 変数の更新**) {
    繰り返し実行される処理
}

例：1から10 までの総和を求める処理.

```
for (i=1;i<11; i=i+1) {
    sum  = sum + i;
}
```

while文

```
while (繰り返し継続条件) {
    繰り返し実行される処理
}
```

例：1から10 までの総和を求める処理.

```
i=1;
while (i<11) {
    sum  = sum + i;
    i=i+1;
}
```

if文

```
if (条件) {
    条件が成り立つ場合の処理
}
else {
    条件が成り立たない場合の処理
}
```

例:

```
if (x==19) {
    y =y+1;
} else {
    y=y-1;
}
```

関数

```
関数名 (引数) {
    処理
    return 値;
}
```

値を返さない場合は，return**文は不要．**

例：

```
plus(a, b) {
    c = a + b;
    return c;
}
```

アルゴリズム1.4(最大値の計算)

入力: *n*個の整数 x[1], x[2],…, x[n]

アルゴリズム:

```
max=x[1];
for (i=2; i<n+1; i=i+1) {
    if(max<x[i]) {
        max=x[i];
    }
}
```

このアルゴリズムの時間計算量はO(1)×(n-1)=O(n).

アルゴリズム1.5 (等しい整数の出力)

入力: *n***個の整数** x[1], x[2],…, x[n]

アルゴリズム:

```
for (i=1; i<n; i=i+1) {
    for (j=i+1; j<n+1; j=j+1) {
        if (x[i] == x[j]) {
            x[i]とx[j]は等しいと出力;
        }
    }
}
```

このアルゴリズムの時間計算量は$O(n^2)$.

宿題

第1章の演習問題の全て．（提出しなくてもよい．巻末の解答例を見て自己採点せよ．）