

第2回のキーワード

1

アルゴリズム関係

- アルゴリズム
- 確率, 期待値
- 再帰
- 線形探索 (linear search)
- 計算量
- 時間計算量
- 空間計算量
- 最大(最悪)計算量
- 平均計算量
- O 記法(オーダー記法)
- $O(n)$

Java関係(プロIIの復習)

- 配列
- クラスの配列
- ポリモーフィズム(多態性)
- equals

配列要素に関する確率

2

- 単純な確率は対称性で考える
 - 何と何が「同様に確からしい」(対称的な)関係か考察する

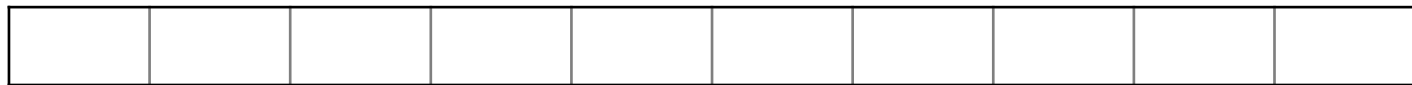
0 1 2 3 4 5 6 7 8 9 の 10通り

完全にランダムなら
どの要素にどの数字が
入るのも立場が同じ

つまり、どれも確率が等しい
(かつ、確率の合計は1)



配列 a



a[0]

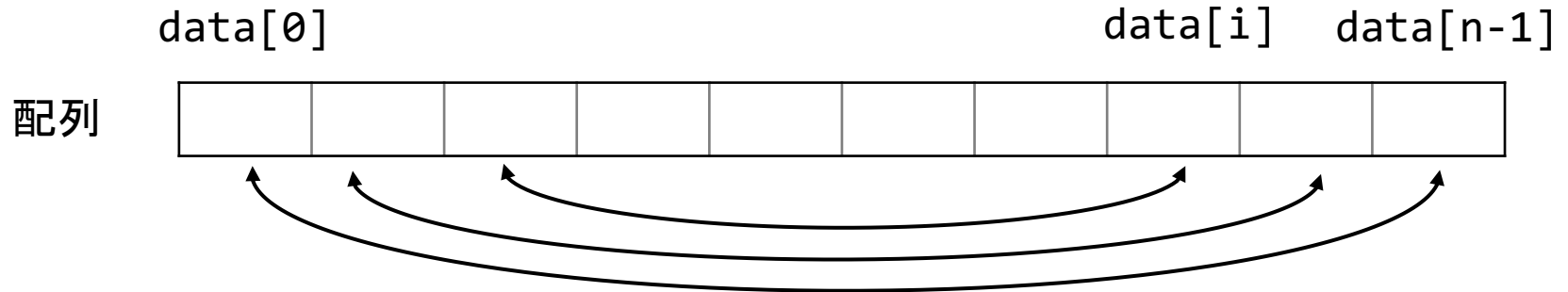
a[i]

a[9]

どのa[i]でも、0~9の特定の数字が入る確率は等しい $\Rightarrow 1/10$

配列の反転のアルゴリズム

3



データ数が n 個の場合, 交換回数は $\lfloor n/2 \rfloor$ 回

```
n = data.length;
for (int i = 0; i < n / 2; i++) {
    double t = data[i];
    data[i] = data[n - 1 - i];
    data[n - 1 - i] = t;
}
```

ソースコードを分析すると
データ数に依存するのは
このループ回数だけ



配列の長さが2倍になると
交換回数も2倍

筆算のアルゴリズム

4

- 筆算と時間の計算は共通の方法
 - ▣ 下の位(桁)から順に計算し, 繰り上がり(繰り下がり)があったら, 1つ上の位に加算(減算)していく
 - ▣ 非常に桁数が多い整数を扱うプログラムにも応用される

	1	7	4	
+		5	9	
	2	3	3	



下の位から上の位へ計算

	18時間	34分	48秒	
+)	9時間	53分	35秒	
1日	4時間	28分	23秒	



下の位から上の位へ計算

アルゴリズムの性能評価

5

- よいアルゴリズムとは？
 - 速い(所要時間) ⇒ 「時間計算量」が少ない
 - 小さい(使用メモリ) ⇒ 「空間計算量」が少ない
 - これらは、両立が難しいことが多い

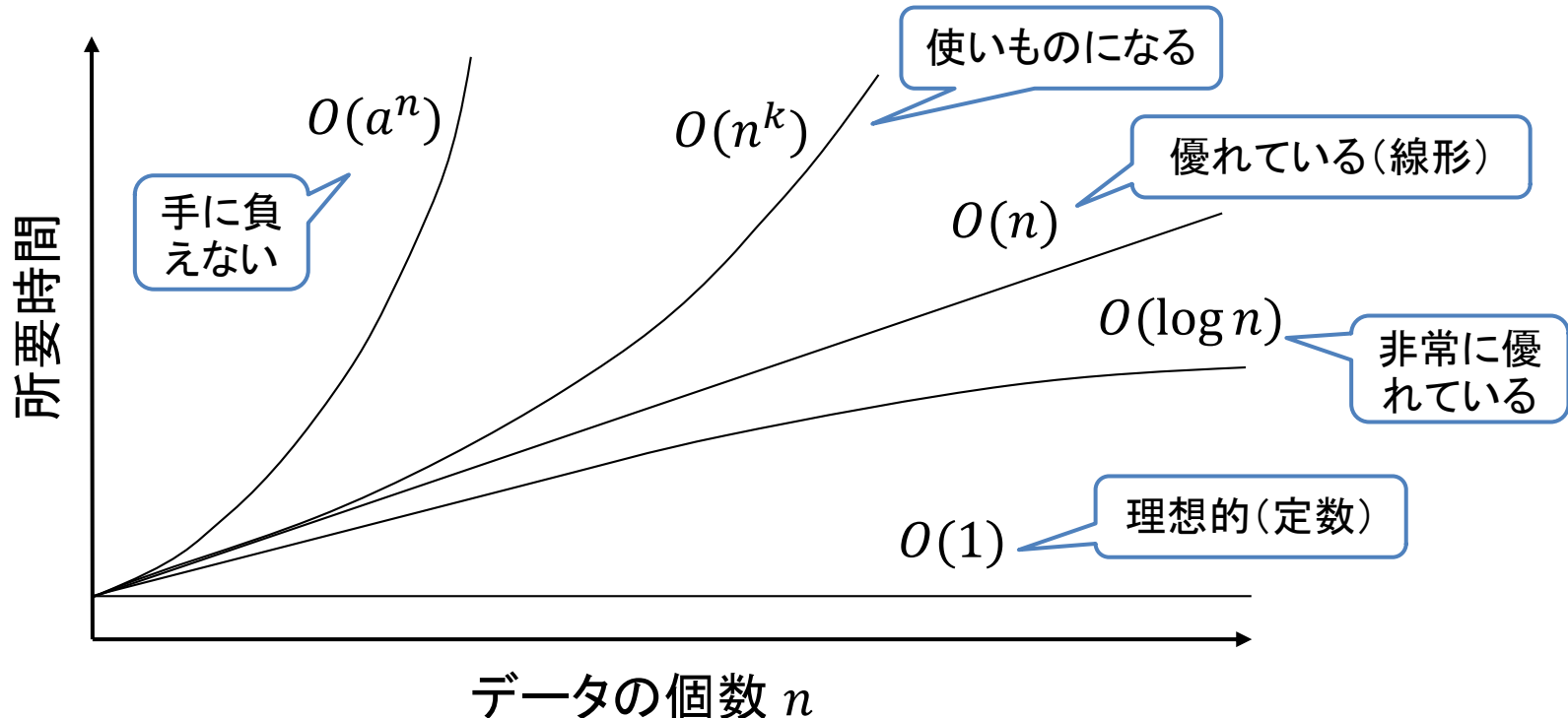
- 計算量の指標
 - アルゴリズムの種類によって、データの並び方や構造に対する得意・不得意がある
 - 最大計算量：そのアルゴリズムが最も不得意なデータセットに対する計算量
 - 例：小さい順に並んでいるデータセットから最大値を線形探索
 - 平均計算量：あらゆるデータセットに対する平均の計算量

オーダー記法

6

□ 計算量の数学的表現

- ほとんどアルゴリズムは、データの個数(n)が増えれば増えるほど、計算に時間がかかるようになり、メモリも必要になる
- n に対する計算量の増加を“関数の種類”(最も影響が大きい関数の定数係数を除いた形式)で分類し、 O 記法で表す



線形探索の考え方

7

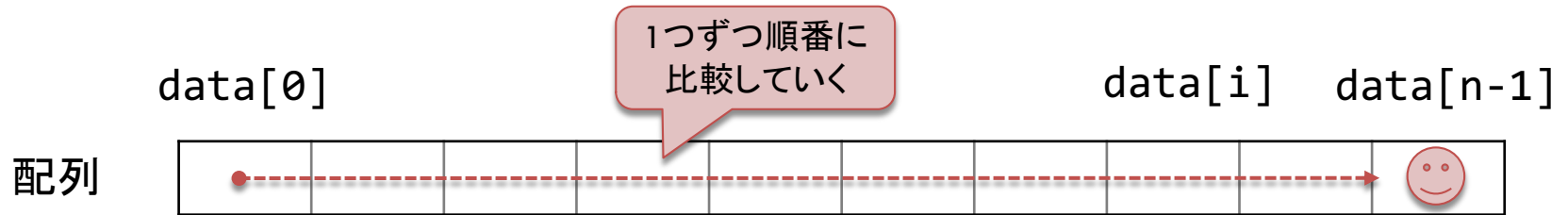
- トランプにたとえると...
 - カードが裏返しに並んでいる(特に順番はない)
 - この中で探したいカードはどこにあるか？



- どうやって探すか？
 - とりあえず、左から(または右から)順にめくってみよう

線形探索の最大計算量

8



データ数は n 個なので、比較回数は n 回

最後に見つかる
(または見つからない)

比較以外にかかる時間は、 n に関わらずほぼ一定(定数)



n が十分大きい場合 ($n \rightarrow \infty$) には比較部分の影響だけ考えればよい



時間計算量は $O(n)$

線形探索の平均計算量

2

1つずつ順番に比較していく

data[0] data[i] data[n-1]

配列

そこにkeyがある確率	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$...	$\frac{1}{n}$...	$\frac{1}{n}$
	×	×	×	×	×	×	...	×	...	×
発見までの比較回数	1	2	3	4	5	6	...	$i+1$...	n
							
比較回数の期待値	$\frac{1}{n}$	$\frac{2}{n}$	$\frac{3}{n}$	$\frac{4}{n}$	$\frac{5}{n}$	$\frac{6}{n}$...	$\frac{i+1}{n}$...	$\frac{n}{n}$

平均計算量

$$= \frac{1}{n} (1 + 2 + \dots + n) = \frac{1}{n} \cdot \frac{n(n+1)}{2} = \frac{n+1}{2} = \frac{1}{2}n + \frac{1}{2}$$

$n \rightarrow \infty$ を考え
 $O(n)$

数列の和の復習

10

□ 等差数列の和

□ 考え方：順番をひっくり返したものを足す

$$\begin{array}{r}
 S = 1 + 2 + \dots + (n-1) + n \\
 +) S = n + (n-1) + \dots + 2 + 1 \\
 \hline
 2S = (n+1) \times n
 \end{array}
 \quad \therefore S = \frac{n(n+1)}{2}$$

□ 等比数列の和

□ 考え方：公比をかけたものと差をとる

$$\begin{array}{r}
 S = 2 + 4 + 8 + \dots + 2^{n-1} + 2^n \\
 -) 2S = \quad 4 + 8 + 16 + \dots + 2^n + 2^{n+1} \\
 \hline
 (1-2)S = 2 - 2^{n+1}
 \end{array}
 \quad \therefore S = \frac{2 - 2^{n+1}}{1 - 2}$$

クラス型の比較

11

- Javaにおけるクラス型の比較
 - ▣ 演算子は使えない (Javaは演算子を多重定義できない)
 - ▣ 代わりに, 以下のメソッドを使う

- `x.equals(y)`
 - ▣ 意味的に $x = y$ ならば `true`, そうでなければ `false` を返す
 - ▣ Javaの全てのクラスがObjectクラスから継承するメソッド

- `x.compareTo(y)`
 - ▣ $x < y$ ならば負, $x = y$ ならば 0, $x > y$ ならば正を返す
 - ▣ StringやIntegerなど, Comparableインタフェースを実装するクラスが持つメソッド