第3回のキーワード

アルゴリズム関係

- 2分探索（binary sarch）
- 再帰
- $O(\log n)$

Java関係

- compareTo

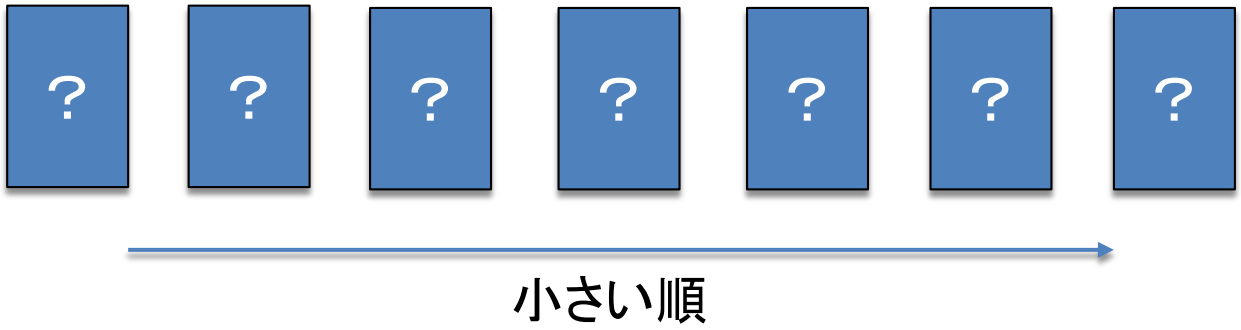
本講義の学修では，演習課題の説明内容もよく読んで理解してください。
この資料は図解が中心なので，細かい部分は説明不足になっています。

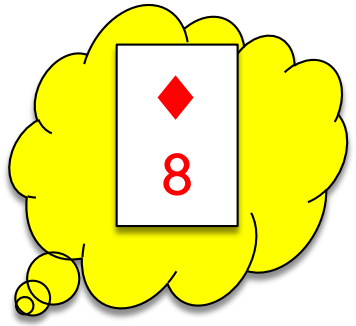
もっと速い探索方法はないの？

- 前回紹介「線形探索」
  - 先頭から順番に調べていくしかないのか？
  - コンピュータなら，1000人の名前がバラバラに並んだ名簿からでも一瞬で検索できるけど…
- データ構造を工夫する
  - もっと効率的に探し出すためには，どうすればいいか？
  - 世の中の情報は，どのように整理してあるだろうか？
- 探索の高速化戦略
  - 整列しておく　⇒ 今回説明
  - 分類しておく　⇒ 似たようなことは次回以降説明

2分探索の考え方

- トランプにたとえると…
  - 裏返しのカードが，数の小さい順に並んでいる
  - この中で探したいカードはどこにあるか？

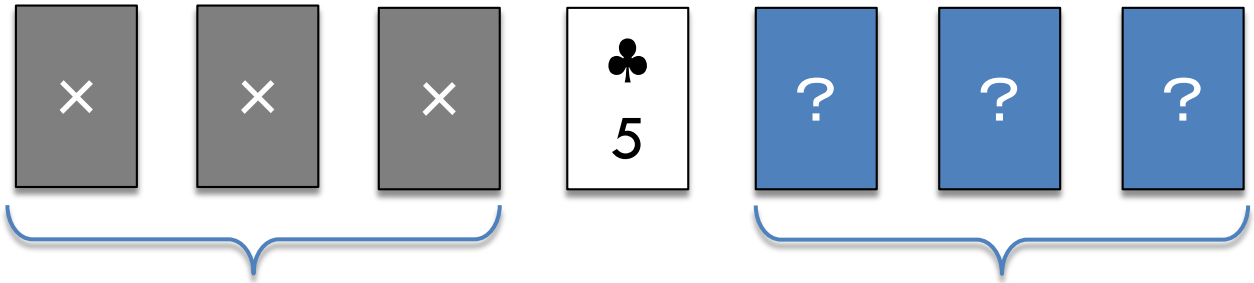




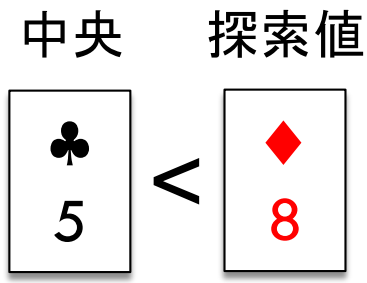
- 戦略を考えてみよう
  - まず，真ん中のカードを開けると何がわかるか？
  - 候補が半分ずつに減っていく（半分→4分の1→8分の1…）

2分探索の考え方

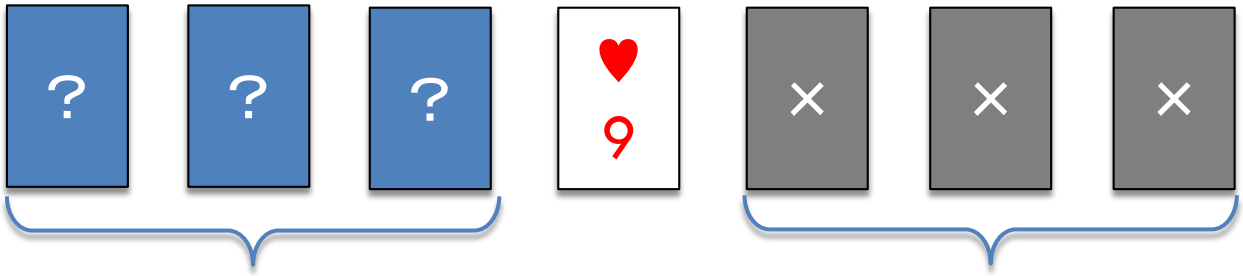
- 中央を開けると探索範囲が半分に絞り込まれる



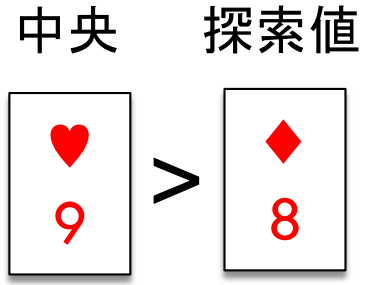
8がある可能性なし　　8がある可能性あり



の場合



8がある可能性あり　　8がある可能性なし



の場合

2分探索のアルゴリズム

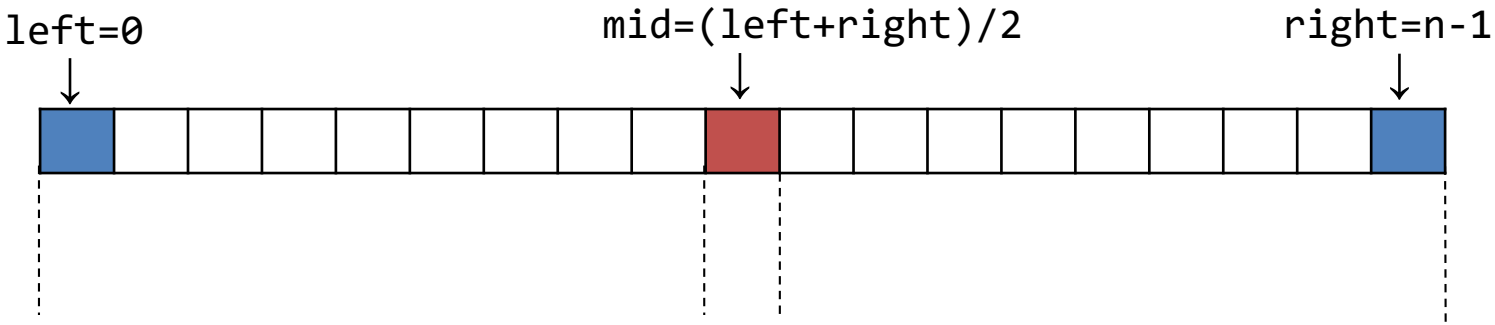
- データ
  - 配列の要素は，小さい順（または大きい順）に並べておく
  - 探索する値をkeyとする

これが高速化のポイント！

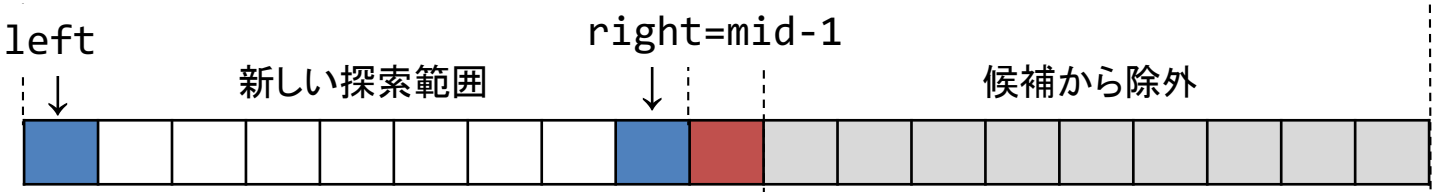
- アルゴリズム
  - 配列（探索範囲）の中央にある値とkeyを比較する
  - もし両者が等しければ，発見したのでその位置を返す
  - もしkeyの方が小さければ，探索範囲を前半分にせばめる
  - もしkeyの方が大きければ，探索範囲を後半分にせばめる
  - 以上の手順を，探索範囲に要素がなくなるまで繰り返す
  - 探索範囲に要素がなくなったら，keyは含まれていない

探索範囲のせばめ方

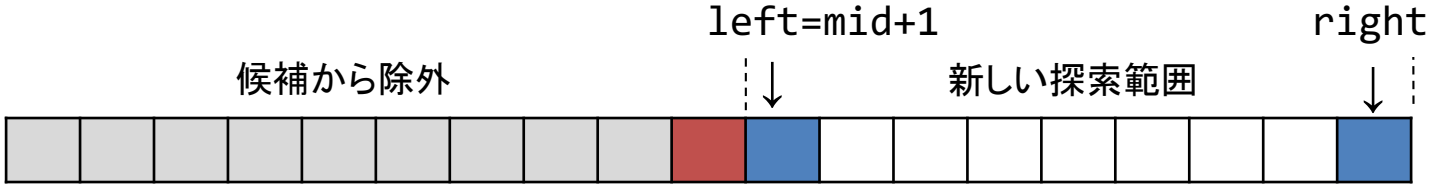
探索範囲の左端と右端を表す変数を使う



左半分にせばめるときは，右端を動かす

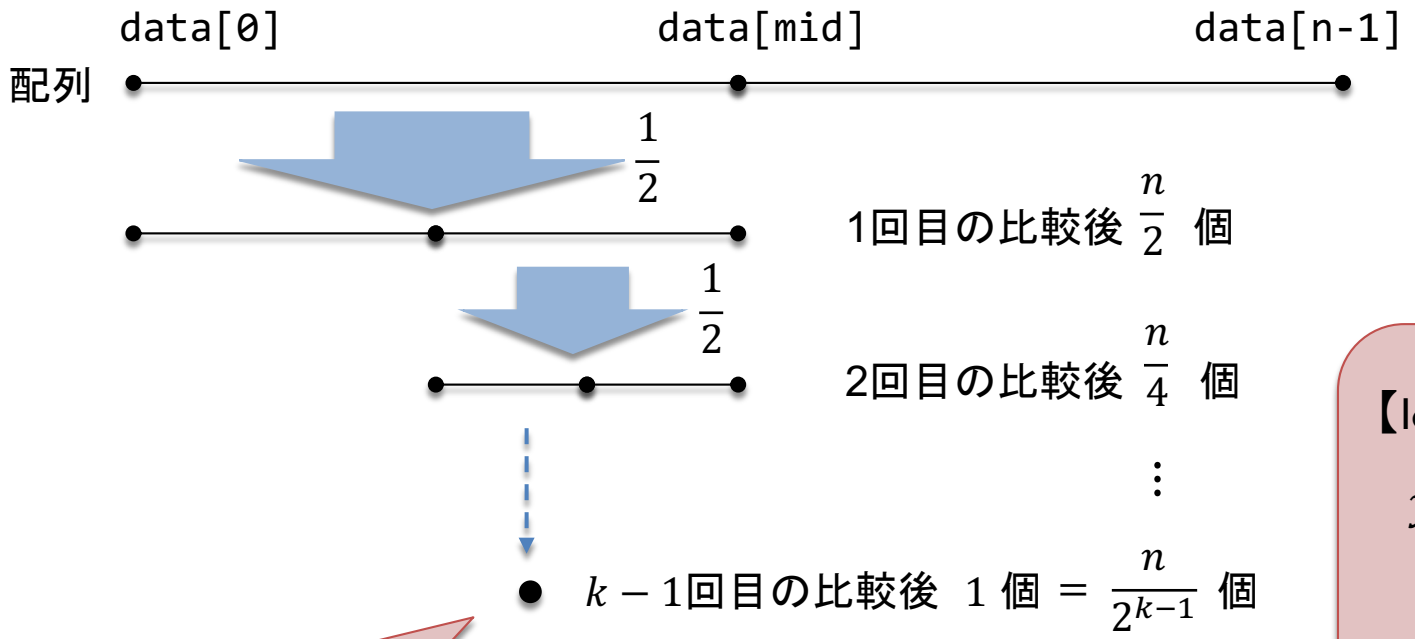


右半分にせばめるときは，左端を動かす



2分探索の最大計算量

総比較回数を仮にk回とおき，kを求める



残りが1個になるのは，最後の比較をする直前だから，$k-1$回目の比較後である

$$\frac{n}{2^{k-1}} = 1$$

$$n = 2^{k-1}$$

$$k = \log_2 n + 1$$

$n \to \infty$ を考え $O(\log n)$

【logの定義】

$y = a^x$ のとき

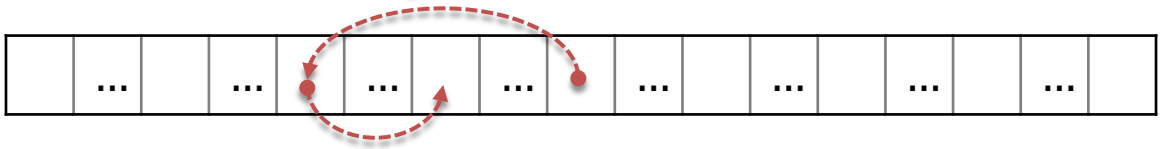
$x = \log_a y$

2分探索の平均計算量

簡単のため $n = 2^k - 1$ とする

中点→“四分”点→“八分”点と比較

data[0] data[mid] data[n-1]



そこにkeyがある確率

$\frac{1}{n}$ $\frac{1}{n}$ $\frac{1}{n}$ $\frac{1}{n}$ … $\frac{1}{n}$ … $\frac{1}{n}$ …

比較回数の期待値

× × × × … × … × …

1回目で発見 1個 1 $= \frac{1}{n} \times 1 \times 1$

2回目で発見 2個 2 2 $= \frac{1}{n} \times 2 \times 2$

3回目で発見 4個 3 3 3 3 $= \frac{1}{n} \times 3 \times 4$

… … … … … … … …

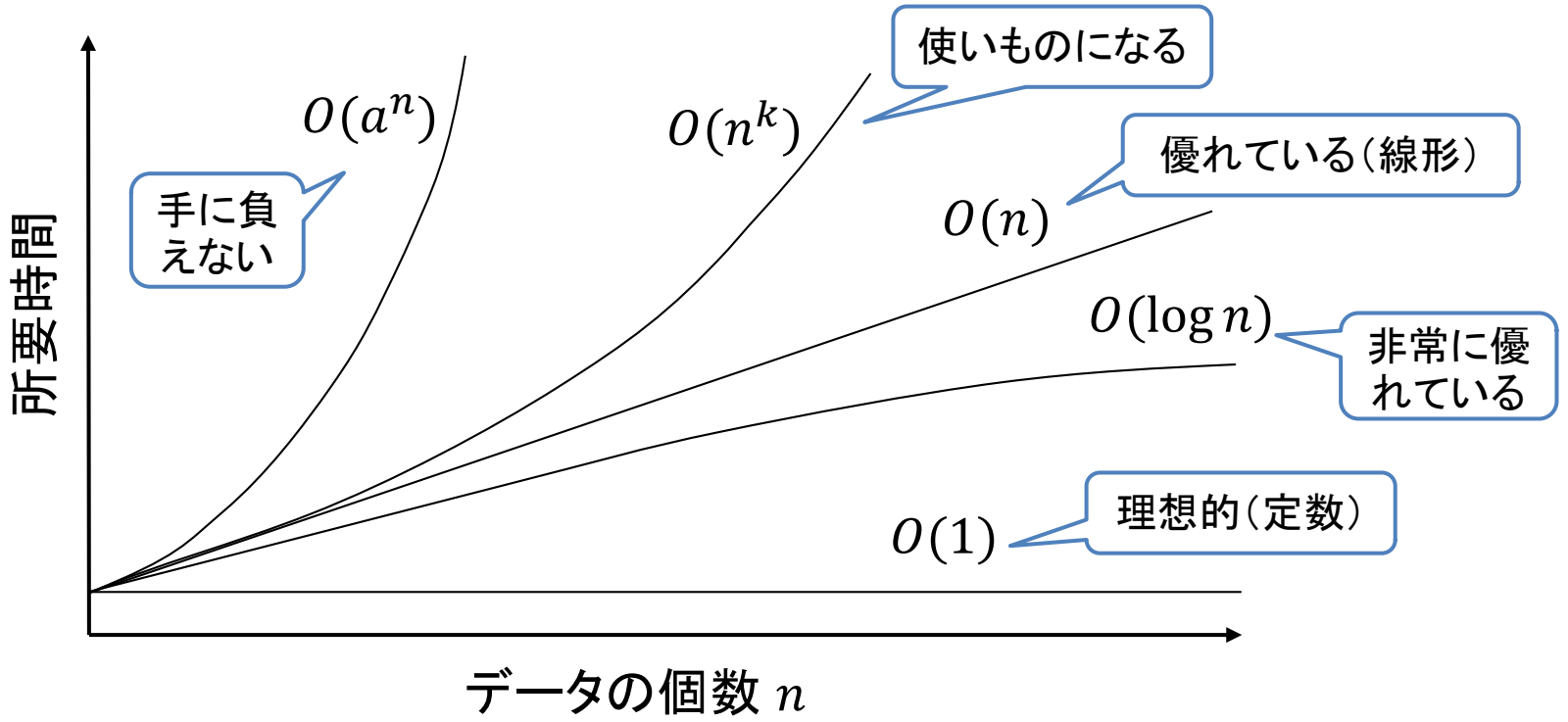
k回目で発見 2^{k-1}個 k k k k k k k k … $= \frac{1}{n} \times k \times 2^{k-1}$ +

さらに $n = 2^k - 1$ を使って n の式にする

$$\frac{1}{n}\left(1 \cdot 1 + 2 \cdot 2 + 3 \cdot 4 + \cdots + k \cdot 2^{k-1}\right)$$

オーダー記法

- 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



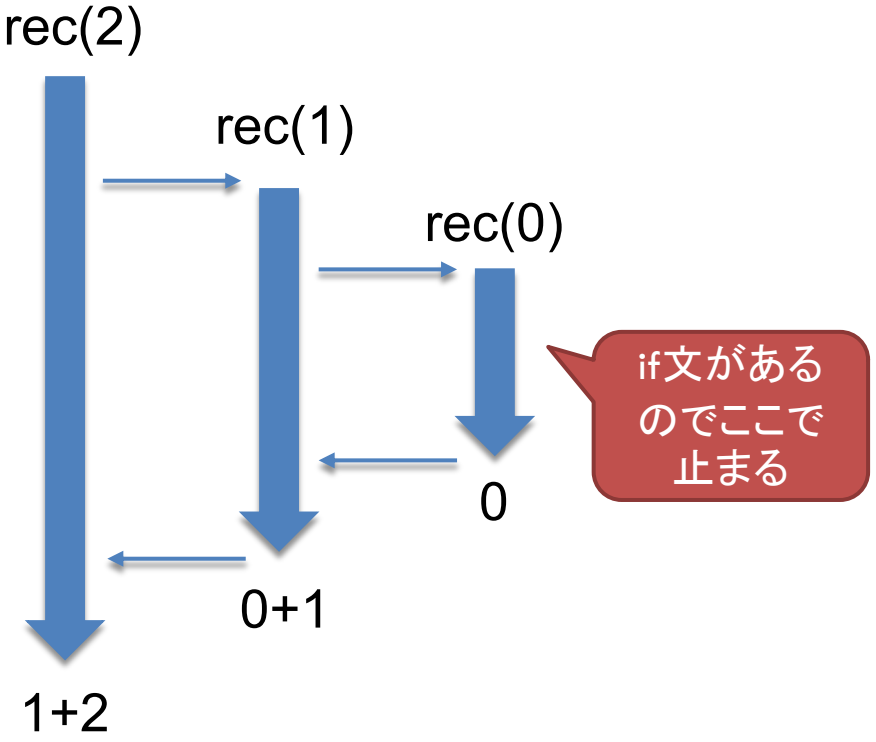
再帰

- 再帰とは
  - 関数（メソッド）が，自分自身（の別のコピー）を呼ぶこと
  - 必ず，再帰の終了判定が必要

```
int rec(int n) {
    System.out.println(n);

     // n=0ならもう再帰しない
    if (n == 0) return 0;

    return rec(n – 1) + n;
}
```



数列の和の復習

- 等差数列の和
  - 考え方：順番をひっくり返したものを足す

$$\begin{aligned} S &= 1 + 2 + \cdots + (n-1) + n \\ +)\ S &= n + (n-1) + \cdots + 2 + 1 \\ \hline 2S &= (n+1) \times n \end{aligned}$$

$$\therefore S = \frac{n(n+1)}{2}$$

- 等比数列の和
  - 考え方：公比をかけたものと差をとる

$$\begin{aligned} S &= 2 + 4 + 8 + \cdots + 2^{n-1} + 2^n \\ -)\ 2S &= \quad 4 + 8 + 16 + \cdots + 2^n + 2^{n+1} \\ \hline (1-2)S &= 2 - 2^{n+1} \end{aligned}$$

$$\therefore S = \frac{2 - 2^{n+1}}{1 - 2}$$

クラス型の比較

- 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