全学共通科目･工学部専門科目

「計算機科学概論」
アルゴリズムとプログラミング
その1

五十嵐 淳
igarashi@kuis.kyoto-u.ac.jp
大学院情報学研究科
通信情報システム専攻

自己紹介(専門分野)

プログラミング言語の研究、特に基礎理論

研究の出発点:

- 自分がうまくプログラムが書けないのを言語のせいにする

- プログラムの間違いを自動発見する仕組みを作る
- そもそも間違いを犯しにくいプログラミング言語を作る

担当分のメニュー

- 6/21, 6/28: アルゴリズムについて
- 7/5: 出張につき休講
- 7/12, 7/19: プログラミングについて
- (補講の予定・内容は未定です)

講義情報

http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/cs-intro/

今日のメニュー

- アルゴリズムとは何か？
- 探索のアルゴリズム
  - 素朴な探索
  - 効率のよい探索：二分探索
  - アルゴリズムの良し悪しとは？
- 整列ゲーム

アルゴリズム(algorithm)とは何か？

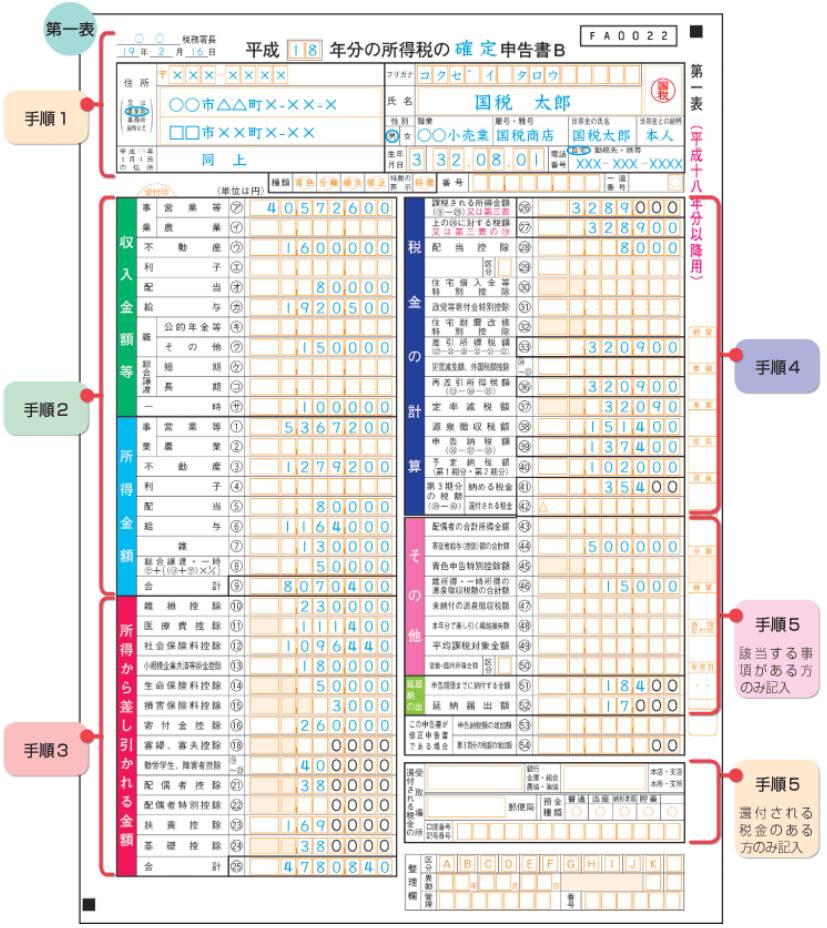
アルゴリズムではないもの

- 「ピ〇ゴ〇スイッチ」の「～こうしん」「～たいそう」

アルゴリズムに近いもの

- 料理のレシピ
- 確定申告書

確定申告書



- 欄によっては他の欄からの計算方法が書いてある
- 例:

再差引所得総額(36)
＝ (33)-(34)-(35)

アルゴリズムとは何か(バージョン1.0)

目的を達成するための停止する手続きの厳密な記述

- 目的 (入力と出力)
  - レシピ: 材料と完成する料理
  - 確定申告書: 年収と税金額
- 厳密な記述

×「材料(2人前): …、塩少々」「味を整えます」

○「塩1gを加えます」

- 停止する手続き

×「野菜を泥がなくなるまで洗います」

○「野菜を泥がなくなるまで洗います。ただし1時間経ってもなくならない場合は諦めて次に進んでください」

某たいそうの
目的は何だろう？

今日のメニュー

- アルゴリズムとは何か？
- 探索のアルゴリズム
  - 素朴な探索
  - 効率のよい探索：二分探索
  - アルゴリズムの良し悪しとは？
- 整列ゲーム

探索とは？

「以下の数の集まりに 758 があるか。YesかNoか?」

682, 643, 223, 328, 494, 171, 216, 947, 398, 525, 395,
159, 267, 38, 191, 508, 860, 815, 39, 545, 28, 595, 160,
802, 995, 444, 151, 10, 230, 470, 529, 289, 790, 471,
918, 324, 586, 860, 928, 43, 19, 623, 120, 807, 262, 648,
308, 334, 762, 866, 303, 396, 106, 530, 727, 991, 397,
945, 953, 954, 106, 357, 174, 464, 338, 915, 500, 571,
595, 454, 467, 518, 331, 175, 833, 313, 271, 506, 726,
85, 996, 459, 276, 900, 458, 235, 654, 106, 145, 829, 218,
306, 385, 796, 365, 502, 929, 878, 623, 341

探索アルゴリズムの目的

入力: ものの集まり(データ集合)ともの(検索キー)

出力: 検索キーがデータ集合の要素かどうか (Yes/No)

変形バージョン

入力: 辞書(見出し語と項目説明の組の集まり)と見出し語

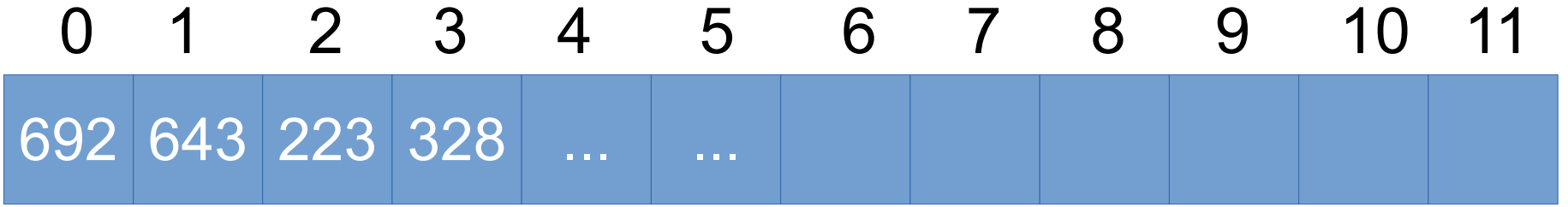
出力: 見出し語に対応する項目説明

データ構造(data structure)

- ものの集まりの「表現方法」
- データ構造ごとに操作の種類が決まっている
- データ構造が変わればアルゴリズムも変わる
  - アルゴリズムの良さも変わる
- 基本的なデータ構造
  - 配列(array)
  - リスト(list)
  - 木(tree)
  - キュー(待ち行列, queue)
  - グラフ(graph)

配列

- データに「背番号」(添字、インデックス)を割り当てる



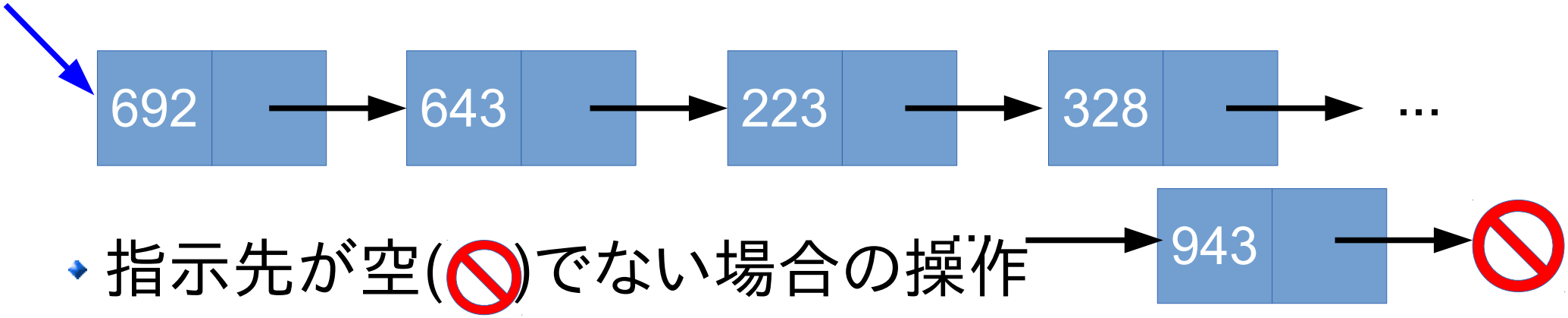
- インデックスを指定しての要素の読み出し、更新操作
  - インデックスによらず一定の時間で読み出し･更新可能
    - 配列 X の *i* 番目の要素を X[*i*] と呼ぶ
  - 配列の長さは(ふつうは)固定されている

配列を使った探索アルゴリズム

1. 入力データの配列の長さを N とする、また $i = 0$ とする
2. 以下の作業を繰り返す
   - $i \geqq N$ なら 3. へ
   - 配列の i 番目の要素を読み出す
   - 検索キーと等しいか？
     - Yes → Yes をアルゴリズム全体の答えとして終了
     - No → i の値を +1 して、2. に戻る
3. No を全体の答えとして終了

リスト

- ひとつひとつの要素に「次は何か」の情報(ポインタ, 指示棒)が付与されている



- 指示先が空(🚫)でない場合の操作
  - 先頭の要素の読み出し
  - 後続リストを指すポインタの取り出し
    - (先頭要素と後続リストの分割)
- 要素の追加･削除が簡単(ポインタのつけかえ)
- 可変長･後ろの要素の読み出しには時間がかかる

リストを使った探索アルゴリズム

1. 入力はリストを指すポインタ
2. 与えられたポインタの指す先は空()か？
   - Yes → No を全体の答えとして終了
   - No → 先頭要素は検索キーと等しいか？
     - Yes → Yes を全体の答えとして終了
     - No → 後続リストへのポインタを新しい入力として探索続行(1.へ)

チェックポイント!

ふたつの「アルゴリズム」の

- 厳密さ
- 停止するか
- 目的を本当に達成するか

を考えてみよう

今日のメニュー

- アルゴリズムとは何か？
- 探索のアルゴリズム
  - 素朴な探索
  - 効率のよい探索: 二分探索
  - アルゴリズムの良し悪しとは？
- 整列ゲーム

入力データの種類を限ると探索は速くできる

「以下の数の集まりは小さい順に並んでいます。この中に 758 があるか。YesかNoか?」

5, 6, 15, 20, 43, 44, 49, 57, 62, 65, 68, 77, 85, 85, 101, 107,
113, 124, 168, 173, 174, 181, 182, 186, 188, 193, 194, 199,
213, 232, 238, 238, 241, 271, 281, 284, 296, 305, 325, 337,
348, 354, 383, 393, 405, 409, 437, 439, 442, 466, 479, 490,
493, 499, 502, 504, 507, 510, 512, 525, 534, 544, 547, 550,
561, 565, 578, 606, 614, 614, 615, 618, 626, 632, 633, 643,
645, 682, 716, 726, 736, 743, 751, 765, 790, 807, 825, 836,
845, 869, 876, 925, 927, 928, 944, 965, 977, 978, 997, 1000

二分探索(binary search)

- まず、真ん中を見る
- 外れでも、小さい順に並んでいるので求めるものがどっち側にないかが確定
  - 半分のデータはもう無視してもよい
  - あるかもしれない方の真ん中を見る
    - 外れでも、小さい順に並んでいるので…
    - …

二分探索アルゴリズム

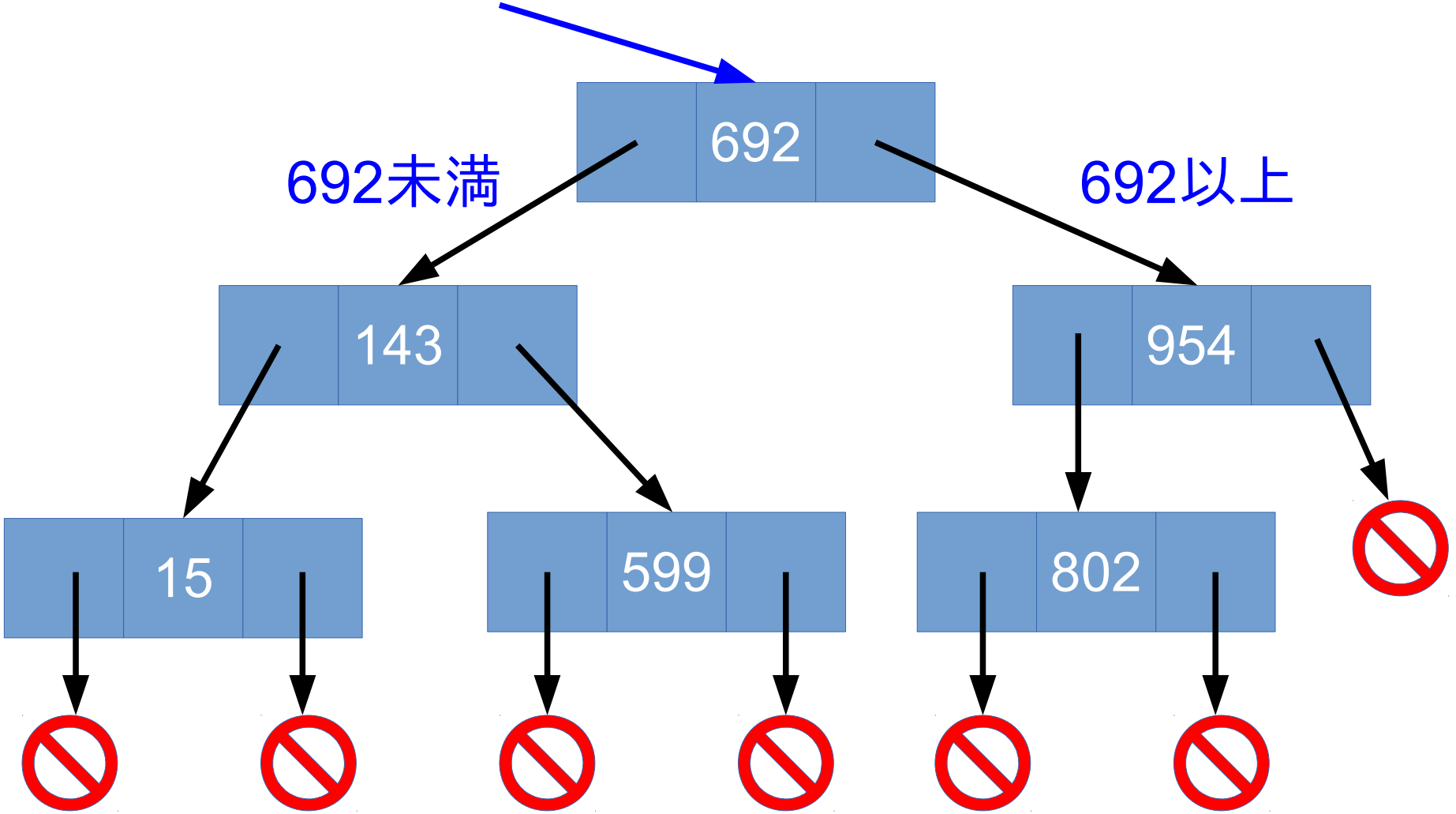
1. 入力データは配列(長さ N)とする
2. 探索範囲を表す変数 $(x, y) = (0, N-1)$ とする
3. x ≦ y である限り以下を繰り返す
   - 配列の i = (x+y)/2 番目(小数点以下切り捨て)の要素 z を読み出し、検索キー k と比較
     - k = z → Yes を全体の答えとして終了
     - k < z → y の新しい値を i -1 として 3.に戻る
     - k > z → x の新しい値を i+1 として 3.に戻る

二分探索の適用条件

- データの集まりは配列で表現されている
  - 真ん中へんのデータをいきなり読み出せる必要
- 順序づけ可能なデータ
  - 数でなくてもよい
  - 辞書の見出し語
- データは予め順序に従って並べられている
  - どうやって並べるの？ → 整列(sorting)

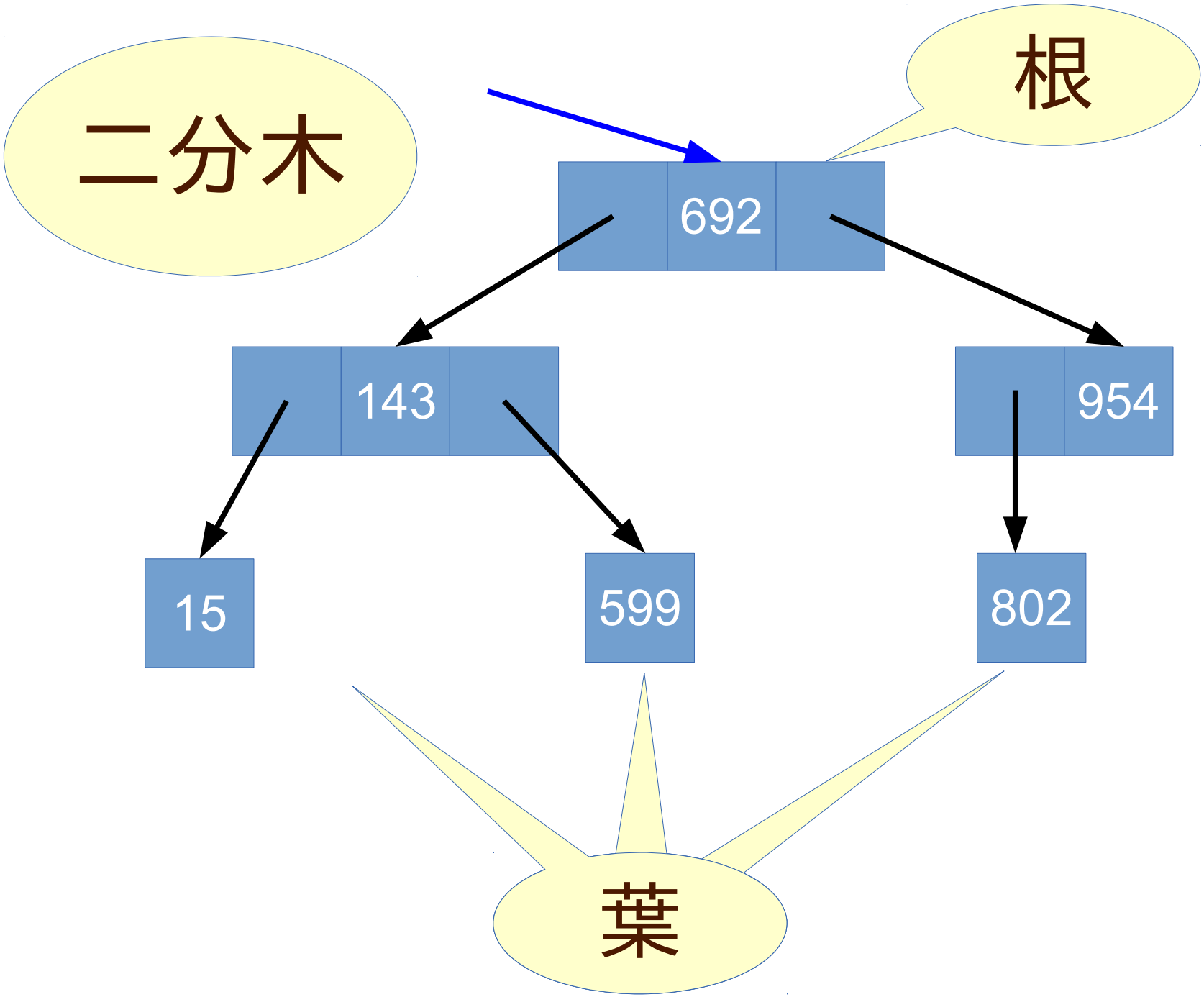
二分探索木(binary search tree)

- ポインタを使った効率のよい探索用データ構造



一般の木構造

- 節点(node)同士がポインタでつながれている
- 根(root)と呼ばれる節点があり全ての節点へポインタを辿って到達可能、辿り方は一種類
  - 計算機科学では、多くの場合根を上に書く
- 葉(leaf)
  - ポインタが出ていない( しか指してない)節点
- n分木(n-ary tree)
  - 節点から出ているポインタ数の最大数が n の時
- 節点 x の部分木(subtree)
  - x から出るポインタの先を根とする木


二分木
根
692
143
954
15
599
802
葉

二分探索木による探索アルゴリズム

1.入力は木の根を指すポインタ

2. 与えられたポインタの指す先は空()か？

- Yes → No を全体の答えとして終了
- No → 根ノードのデータ x と検索キー y を比較
  - x = y → Yes を全体の答えとして終了
  - x < y → 右の部分木へのポインタを新しい入力として探索続行(1.へ)
  - x > y → 左の部分木へのポインタを新しい入力として探索続行(1.へ)

今日のメニュー

- アルゴリズムとは何か？
- 探索のアルゴリズム
  - 素朴な探索
  - 効率のよい探索: 二分探索
  - アルゴリズムの良し悪し･効率とは?
- 整列ゲーム

アルゴリズムの良し悪しの基準

- 所定の目的を達成する(当然?)
  - アルゴリズムの正しさの証明
- 速い
  - 時間計算量(time complexity)が小さい
- メモリ使用量が少ない
  - 空間計算量(space complexity)が小さい

時間計算量の見積り

- ハードウェア・コンパイラによらない「時間」の定義

  × 秒数…コンピュータが速くなれば短くなる

  △ コンピュータが実際実行した命令数

  ○ アルゴリズム記述レベルでの基本操作の数

    - 基本操作…条件判断、配列読み出し、etc.
- 入力が変わればかかる時間も変わる

  ⇒ 入力の大きさ(探索ならデータの数)の関数で表す
- 大きさが同じでもかかる時間は変わる

  ⇒ 最悪や平均時間を考える

素朴な探索アルゴリズム(配列版)の(最悪)時間計算量

- $i = 0$ で1ステップ(これは入力に依らない)
- 配列からの読み出し、比較(それぞれ1ステップ)の回数:
  - 答えが Yes → (要素が見つかった位置 + 1)回
    - 最悪 N 回
  - 答えが No → N 回

線形時間アルゴリズム

最悪時間計算量は $2N + 1$

- 数え方にもよるが定数 c, d について $cN + d$ になる

二分探索アルゴリズムの(最悪)時間計算量

- (x, y) = (0, N-1)で2ステップ(これは入力に依らない)
- 配列からの読み出し、比較(それぞれ1ステップ)の回数:
  - 探索範囲は半分、半分になっていくので $\log_2 n$ 回

最悪時間計算量は $c \cdot \log_2 n + d$

対数時間アルゴリズム

O記法と漸近的な評価

係数や定数項は(数え方の詳細によるので)大事でない

- 「データの量(の対数)に比例して探索時間が増える」ことがわかるのが大事

⇒$O(n)$, $O(\log n)$ と表記、「オーダーnのアルゴリズム」「オーダー log n のアルゴリズム」という

$f(n) = O(g(n))$ の定義：
ある m, c があって、任意の $n > m$ について
$$f(n) \leqq c \cdot g(n)$$

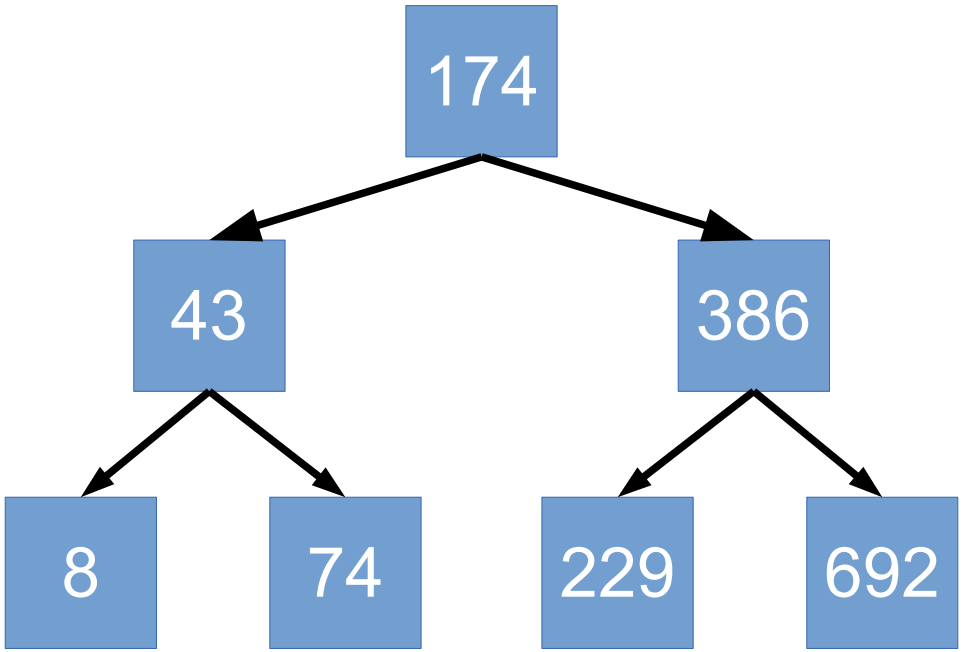
n を十分大きくすれば f は g の定数倍でおさえられ

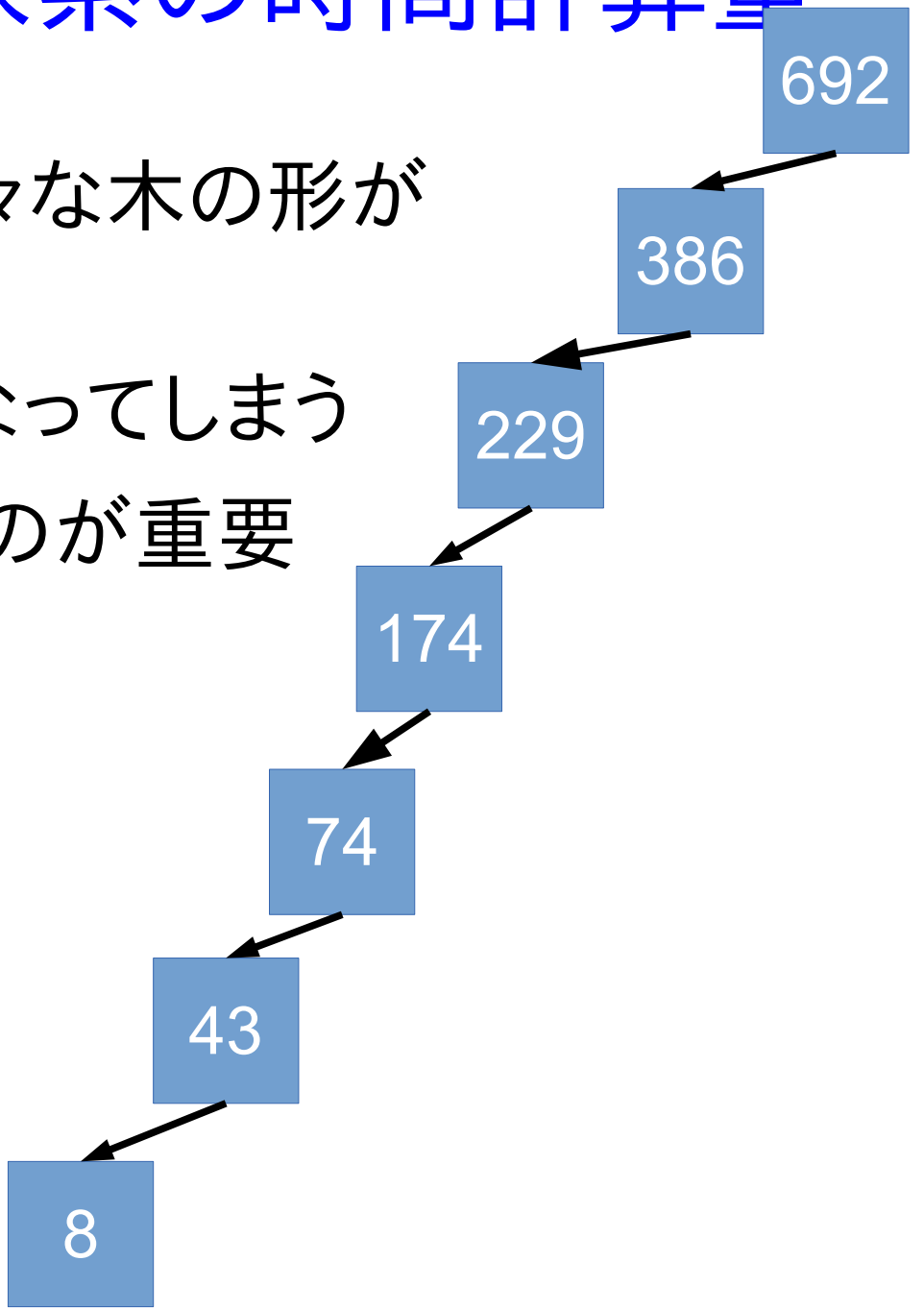
線形 vs 対数

n	$\log n$
2	1
4	2
8	3
16	4
64	6
256	8
1024	10
8092	13
65536	16
1048576	20

二分探索木による探索の時間計算量

- 同じデータの集まりでも色々な木の形がありえる
- 木の形によってはリストになってしまう
- バランスのとれた木を作るのが重要





アルゴリズムの良し悪し： まとめ

- 入力サイズに関して平均･最悪の「時間」を考える
  - (ハードウェアの種類によらない)抽象的な「時間」
  - (空間計算量も基本的な考えは同じ)
- 線形時間アルゴリズムと対数時間アルゴリズム
- 計算量の漸近的評価と O 記法

今日のメニュー

- アルゴリズムとは何か？
- 探索のアルゴリズム
  - 素朴な探索
  - 効率のよい探索: 二分探索
  - アルゴリズムの良し悪し･効率とは？
- 整列ゲーム

整列(sorting)問題

- 入力 X: データの集まり
  - データには順序(全順序)がついている
  - 単純のためデータは相異なるとする
- 出力 Y: 入力データを順序に従って並びかえたもの
  - 配列の場合
    $\forall i$ s.t. $0 \leqq i < N-1$, $Y[i] \leqq Y[i+1]$
    $\forall i$ s.t. $0 \leqq i < N-1$, $\exists$ j, $X[i] = Y[j]$

整列ゲーム

- 登場人物：プレイヤーとマスター、伏せられたトランプ
- 目的：プレイヤーはマスターのヒントをもとに伏せられたトランプを数の小さい順に並べる

プレイヤーに許された行動

- ヒントをもらう：
  - 2枚のトランプを指定しマスターに大小を尋ねる
- カードの交換：
  - 2枚のトランプを指定しマスターに位置を入れ替えてもらう
- 完了宣言：
  - トランプを裏返し目的が達成されたかチェック

参考図書

計算機科学・情報学全体について

- 川合慧(編)、東京大学教養学部テキスト「情報」、東京大学出版会、2006年
- Brian W. Kernighan(著)、久野靖(訳)、ディジタル作法～カーニハン先生の「情報」教室、オーム社、2013年

アルゴリズムについて

- 石畑清(著)、アルゴリズムとデータ構造、岩波書店、1989年