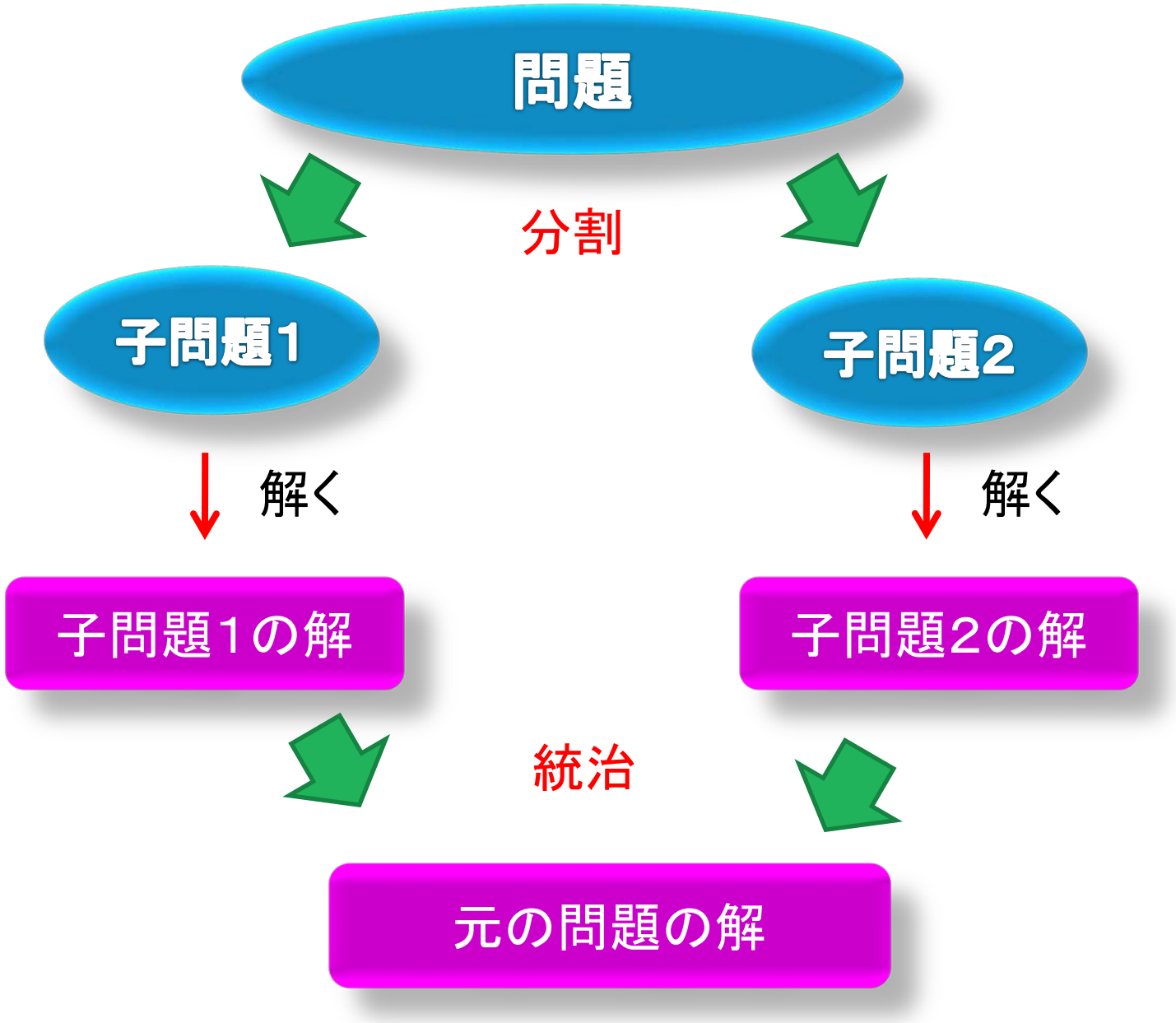
アルゴリズム入門(3)
(分割統治法)

宮崎修一
京都大学　学術情報メディアセンター

分割統治法とは



ソーティング

ソーティング（並べ替え）
　入力：n個の正の整数
　出力：小さい順に並べ替えたもの

入力　10, 9, 2, 6, 4, 1, 8, 3
↓
出力　1, 2, 3, 4, 6, 8, 9, 10

ソーティングに対するアルゴリズムの例：

アルゴリズム1

n個の数を並べる全ての組み合わせに対して、
それが小さい順に並んでいたら出力。

動作例：
入力　5，1，4，3，2，6

5，1，4，3，2，6　…　×
5，1，4，3，6，2　…　×
5，1，4，2，3，6　…　×
5，1，4，2，6，3　…　×
5，1，4，6，2，3　…　×
.
.
.
1，2，3，4，5，6　…　○

入力がn個のとき、
チェックすべき組み合わせは
$n!$ 通りある。

アルゴリズム2

n個の中から一番小さい数を選び、先頭へ。
n-1個の中から一番小さい数を選び、先頭へ。
n-2個の中から一番小さい数を選び、先頭へ。
⋮
2個の中から一番小さい数を選び、先頭へ。

動作例:

入力 5, 1, 4, 3, 2, 6

1, 5, 4, 3, 2, 6

1, 2, 5, 4, 3, 6

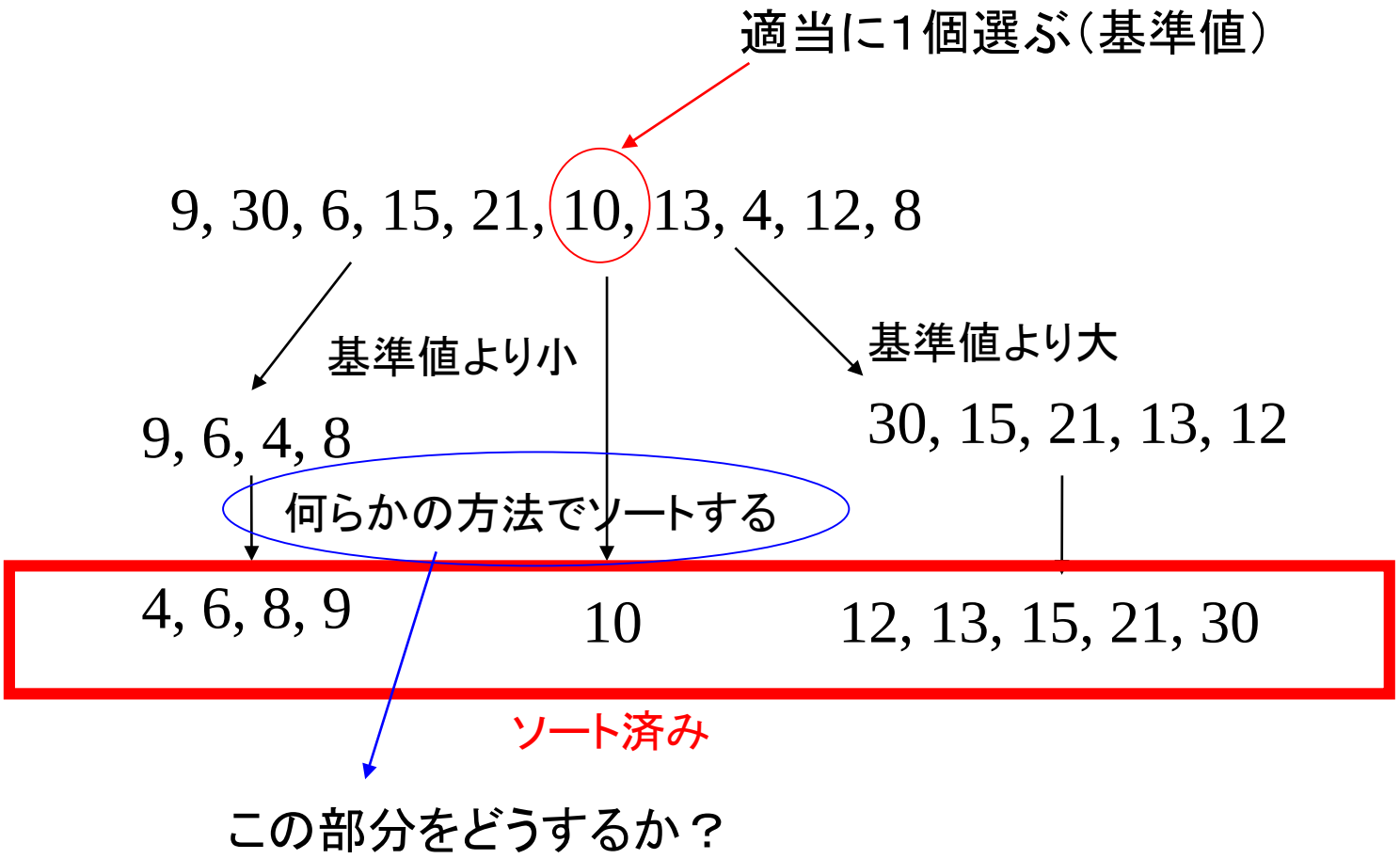
1, 2, 3, 5, 4, 6

⋮

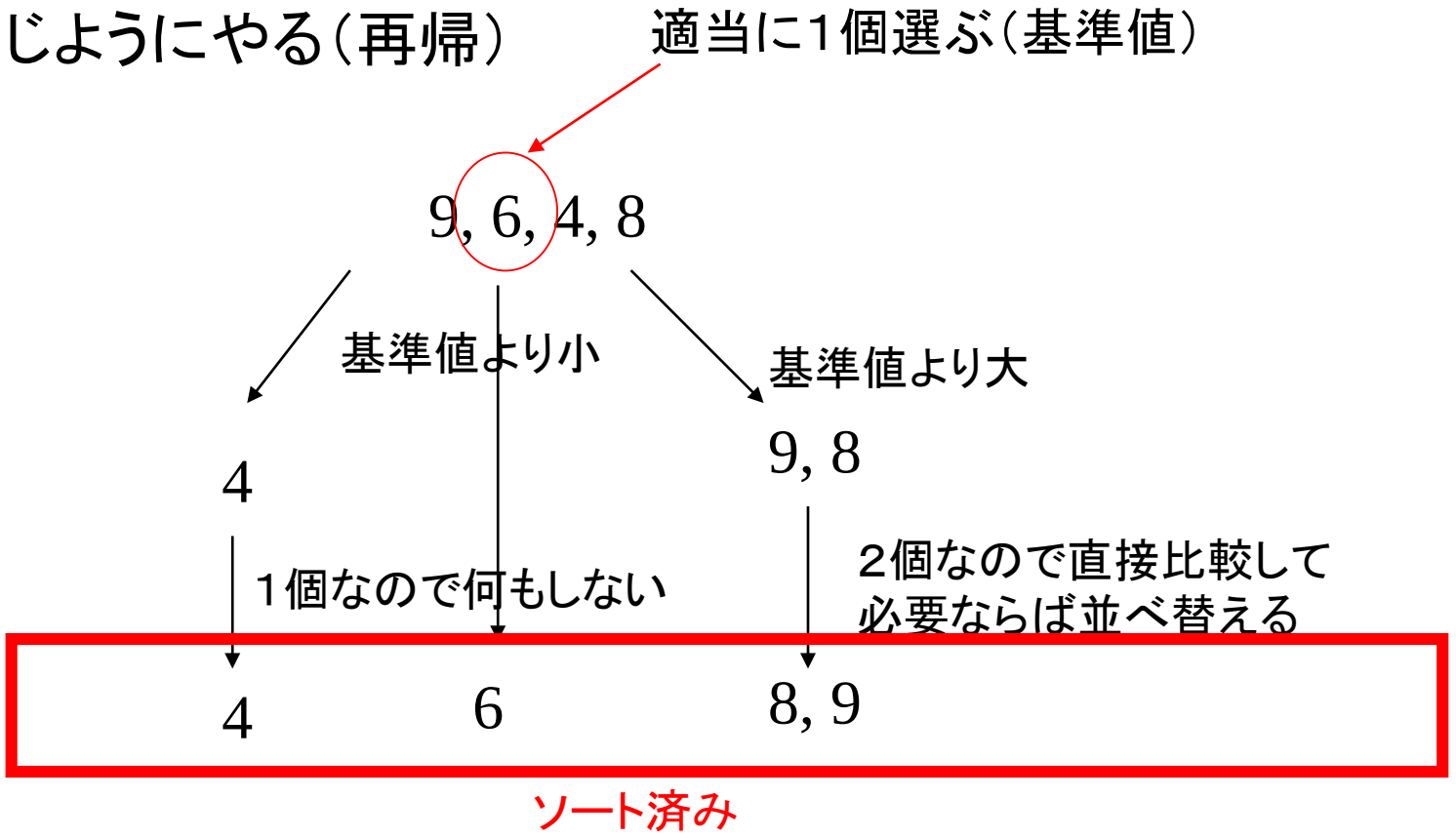
入力がn個のとき、
1回の走査でn個以下チェック
走査はn回やれば十分
↓
全体でn^2回以下

n^2より高速なアルゴリズムはあるだろうか？

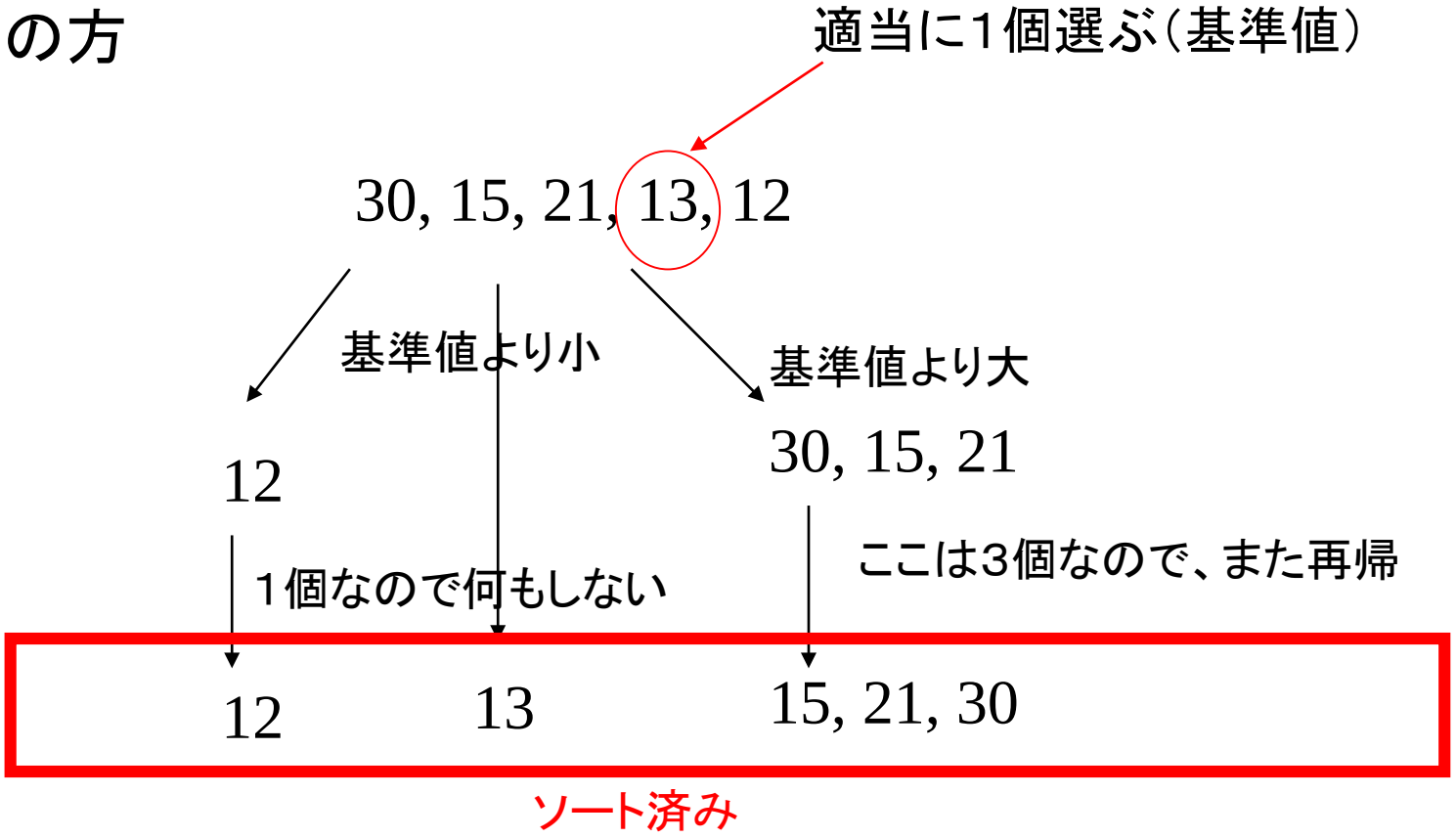
クイックソート



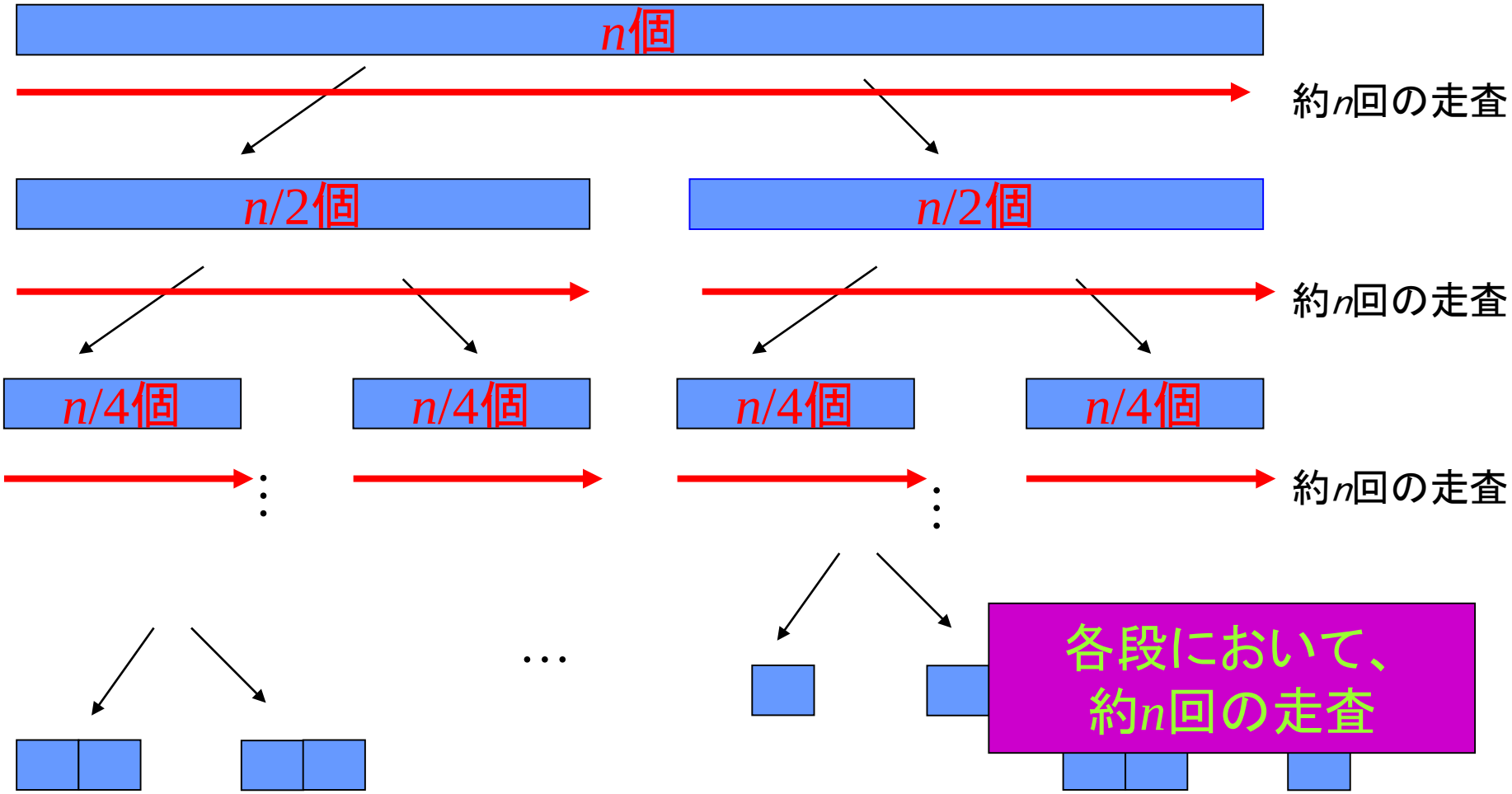
同じようにやる（再帰）


適当に1個選ぶ（基準値）
9, 6, 4, 8
基準値より小
基準値より大
4
9, 8
1個なので何もしない
2個なので直接比較して
必要ならば並べ替える
4
6
8, 9
ソート済み

右の方

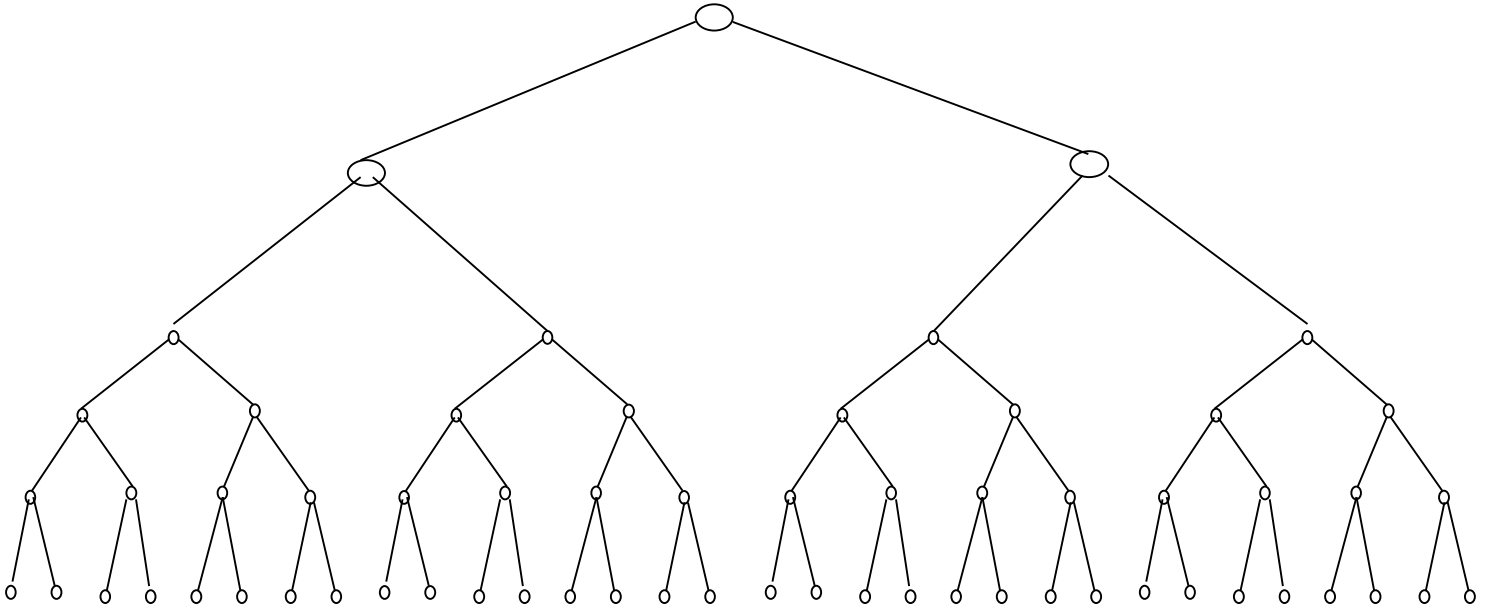

適当に1個選ぶ（基準値）
30, 15, 21, 13, 12
基準値より小
基準値より大
12
30, 15, 21
1個なので何もしない
ここは3個なので、また再帰
12
13
15, 21, 30
ソート済み

計算時間
（毎回半分ずつにデータが分かれたと仮定して）



１個や２個になったら、これ以上分解しない

何段あるか？



$\log_2 n$ 段

n個

計算時間

・1段で、約n回のスキャン

・段数は$\log n$ 段

全体で$O(n \log n)$ 時間　　アルゴリズム2の n^2 から改善

n	2	16	1024	1048576
$\log_2 n$	1	4	10	20

ところが。。。ある（都合の良い）仮定を置いていた。

→「毎回データが半分になる。」

問題：そうならなかった場合、最悪の場合は、計算時間はどうなるだろうか？

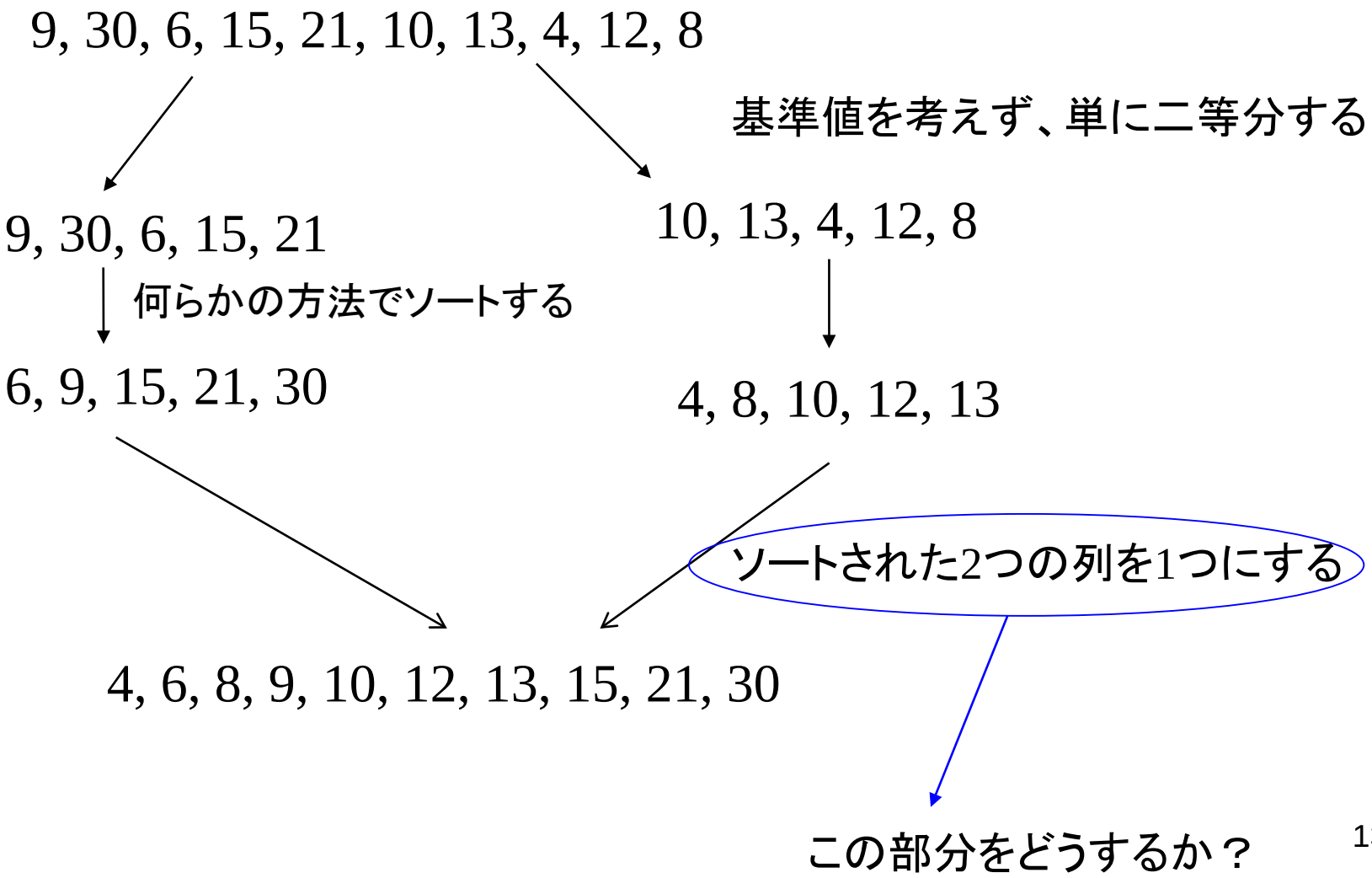
ただし、基準値を一様ランダムに選ぶ場合には、
計算時間の期待値≒$1.39\, n \log n$ であることが証明されている。

つまり、クイックソートは
最悪計算量：$O(n^2)$
平均計算量：$O(n \log n)$

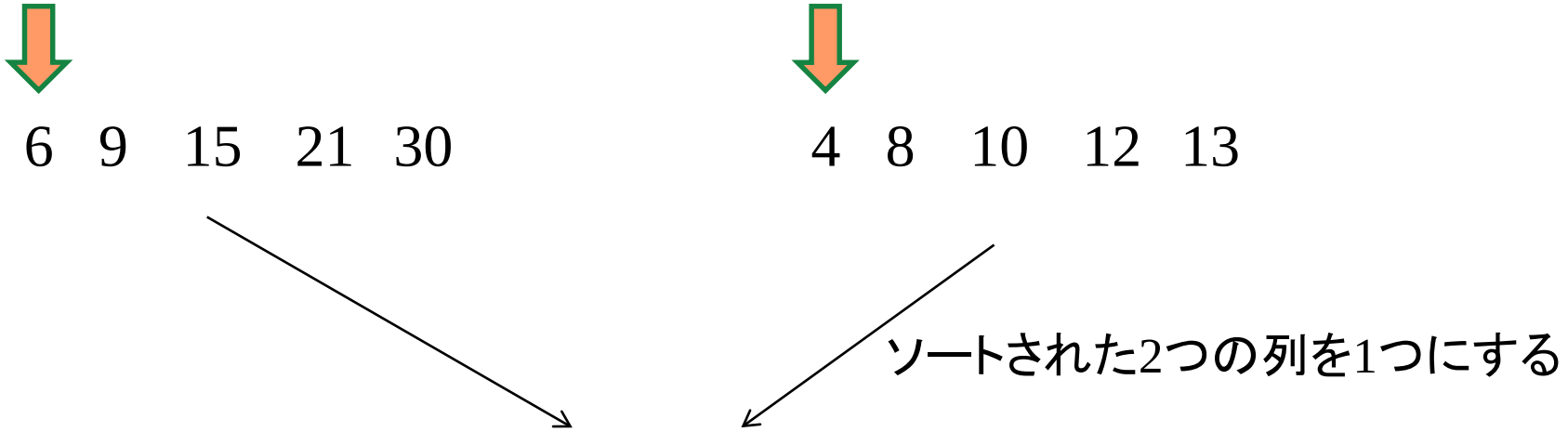
最悪計算量が$O(n \log n)$ のソーティングアルゴリズムもある
- マージソート（これも分割統治法の一種）
- ヒープソート

$O(n \log n)$より速いアルゴリズムが存在しないことも
証明されている。

マージソート


9, 30, 6, 15, 21, 10, 13, 4, 12, 8
基準値を考えず、単に二等分する
9, 30, 6, 15, 21
10, 13, 4, 12, 8
何らかの方法でソートする
6, 9, 15, 21, 30
4, 8, 10, 12, 13
ソートされた2つの列を1つにする
4, 6, 8, 9, 10, 12, 13, 15, 21, 30
この部分をどうするか？

常に列の先頭だけを見る



列の長さに比例した時間O(n)で出来る

データが毎回半分ずつになっているので、
全体の計算量はO($n \log n$)となる。

クイックソートは、分割するときに工夫しているので、統治のときにはあまり手間がかからない。
逆に、マージソートは分割するときに手間をかけず、統治のとき工夫している。

再帰の考え方

自分を記述するために、自分自身を使う

例えばさきほどの、クイックソートの例

QuickSort(I) : 入力列Iをソートする

QuickSort(I)
Iが3個以上の要素からなる場合
{Iの中から、要素を1つ適当に選び、それをxとする。
Iの中で、xより小さな値からなる入力列をAとする。
Iの中で、xより大きな値からなる入力列をBとする。
QuickSort(A)を実行し、その結果をA'とする。
QuickSort(B)を実行し、その結果をB'とする。
A' x B' を出力する。}
Iが2個以下の要素からなる場合
{大小比較して並べ替えて出力する。}

QuickSortというアルゴリズムを記述するために
QuickSortを使っている

その他の再帰の例

階乗の定義

再帰を使わない定義

$n! = n(n-1)(n-2) \cdots 2\ 1$

再起を使った定義

$n! = n \cdot (n-1)!$ （$n \geqq 2$の場合）

$= 1$ （$n = 1$の場合）

$5! = 5 \cdot 4!$

$= 5 \cdot 4 \cdot 3!$

$= 5 \cdot 4 \cdot 3 \cdot 2!$

$= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1!$

$= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$

ハノイの塔