システム工学特講第1

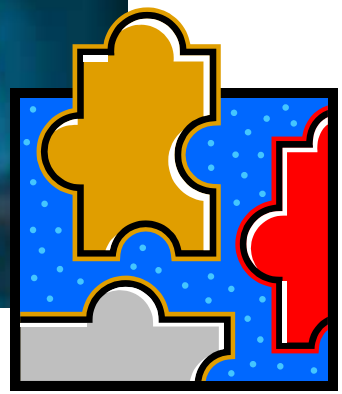
アルゴリズムの比較

効率の良い解法と悪い解法

アルゴリズム

条件
有限長の指示
＋
有限時間で
停止の保障

問題を解くための『手順』



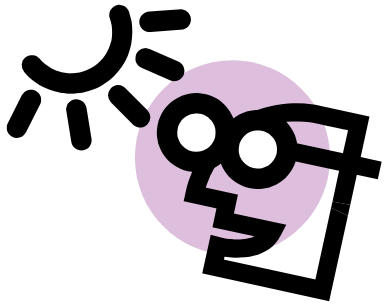
ひとつの問題に対してアルゴリズムは多数

プログラム：
計算機が実行できる指令の系列

ここで学ぶこと

ある問題を解く手順（アルゴリズム）は複数ある

- 「良い」・「悪い」の比較基準は？
  ⇒使用場面で様々な基準が考えられる
  ⇒ここでは，計算時間に注目！
- アルゴリズムの効率を計る方法（計算量）
- アルゴリズムの比較方法（漸近計算量）



準備運動：多項式の値を求める

- **関数** $f(x)=5x^4+6x^3+2x^2+4x+7$

多項式
- x^n **:項**

- $x=3$**の時の** $f(x)$**の値を求めてみよう．**

- $f(x)$**の値を求めるのに，**
  掛け算(×)・足し算（＋）は何回実行した？

- **掛け算・足し算を1回実行するには**
  1円のコストがかかるとする．
  最も安く， $f(x)$**の値を求める方法を提案せよ．**

$f(3)=5\cdot3^4+6\cdot3^3+2\cdot3^2+4\cdot3+7$の計算

直接計算

方法A

- □X=5×3×3×3×3
- □Y=6×3×3×3
- □Z=2×3×3
- □W=4×3
- □$f(3)$=X+Y+Z+W+7

⇒掛け算：10回，
　足し算：4回

変形

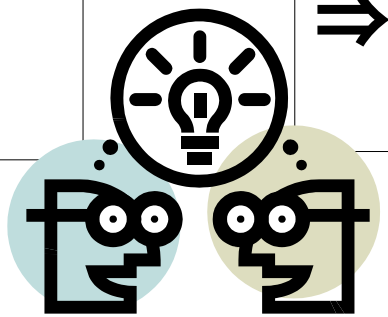
$f(3)=((((5\times3+6)\times3+2)\times3+4)\times3+7)$

計算

方法B

- S=5×3＋6
- T=S×3＋2
- U=T×3＋4
- $f(3)$=U×3+7

⇒掛け算：4回，
　足し算：4回



Horner法

演習2-1

一般のn次多項式

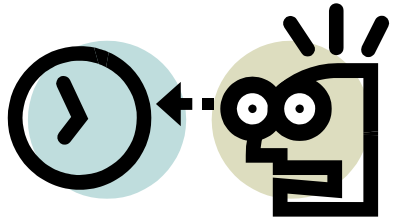
$$f(x)=a_n x^n+a_{n-1}x^{n-1}+\ldots+a_1 x+a_0$$

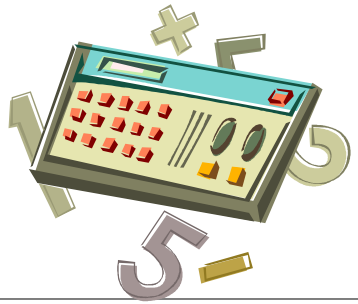
の値を計算したい.

① 方法A利用時の演算総数をnで表せ.

② 方法B利用時は？

③ 1億回/秒演算できるコンピューターがある. n=1,000,000のとき, 各方法では何秒で値を求めるか？





演習2-1：ヒント

方法A

- $X_n = a_n \times \underbrace{x \times \cdots \times x}_{n個}$ 　掛け算：n回
- $X_{n-1} = a_{n-1} \times \underbrace{x \times \cdots \times x}_{n-1個}$ 　掛け算：n-1回
- …
- $X_1 = a_1 \times x$ 　掛け算：1回
- $f(x) = X_n + X_{n-1} + \cdots + X_1 + a_0$ 　足し算：n回

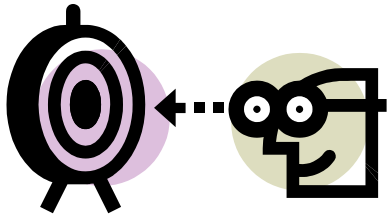
方法B

掛け算：1回，足し算：1回

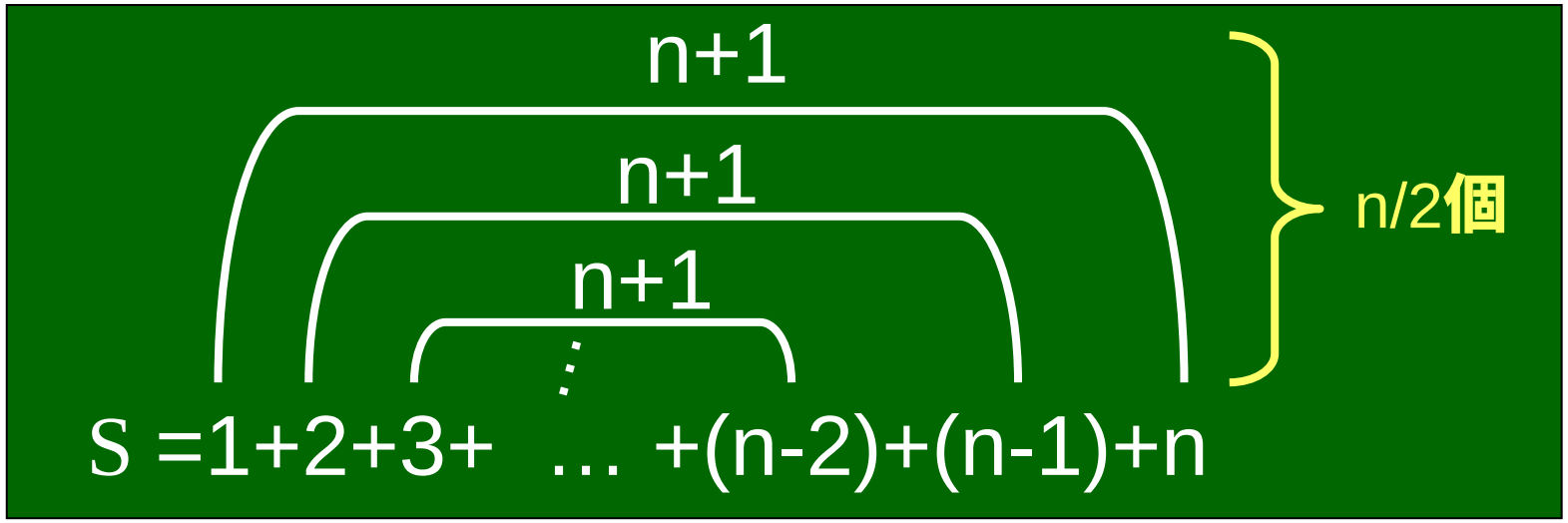
- $S_n = a_n \times x + a_{n-1}$
- $S_{n-1} = S_n \times x + a_{n-2}$
- …
- $S_2 = S_3 \times x + a_1$
- $f(x) = S_1 = S_2 \times x + a_0$

n回の繰り返し

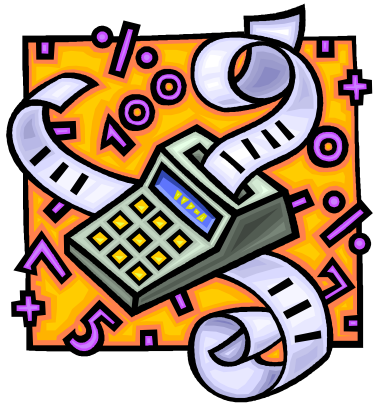
復習 総和を求める(1)



1+2+…+(n-1)+n の求め方



よって, S=(n+1)×(n/2)=n(n+1)/2



演習2-1：例解

方法A

掛け算：n回

- $X_n = a_n \times \underbrace{x \times \cdots \times x}_{n個}$

掛け算：n-1回

- $X_{n-1} = a_{n-1} \times \underbrace{x \times \cdots \times x}_{n-1個}$
- …

掛け算：1回

- $X_1 = a_1 \times x$

足し算：n回

- $f(x) = X_n + X_{n-1} + \cdots + X_1 + a_0$

方法B

掛け算：1回,足し算：1回

- $S_n = a_n \times x + a_{n-1}$
- $S_{n-1} = S_n \times x + a_{n-2}$
- …
- $S_2 = S_3 \times x + a_1$
- $f(x) = S_1 = S_2 \times x + a_0$

n回の繰り返し

総数: $(n+1)n/2 + n$ **回**

どちらが大きい？

総数: $2n$ **回**

アルゴリズムの良し・悪しを比較

- 土俵の整備：**計算モデル**
- ものさしの準備：**計算量**
- 目盛りの読み方：**漸近計算量**

計算の単位

2^{10}の計算
何回押す？

安い電卓の場合

2 * 2 * … 2 =

10+9+1=20回

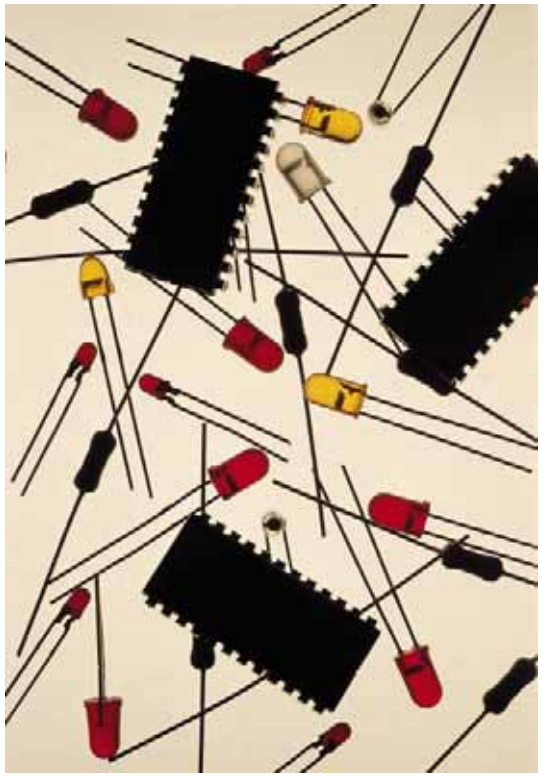
関数電卓の場合

2 x^y 1 0 =

5回

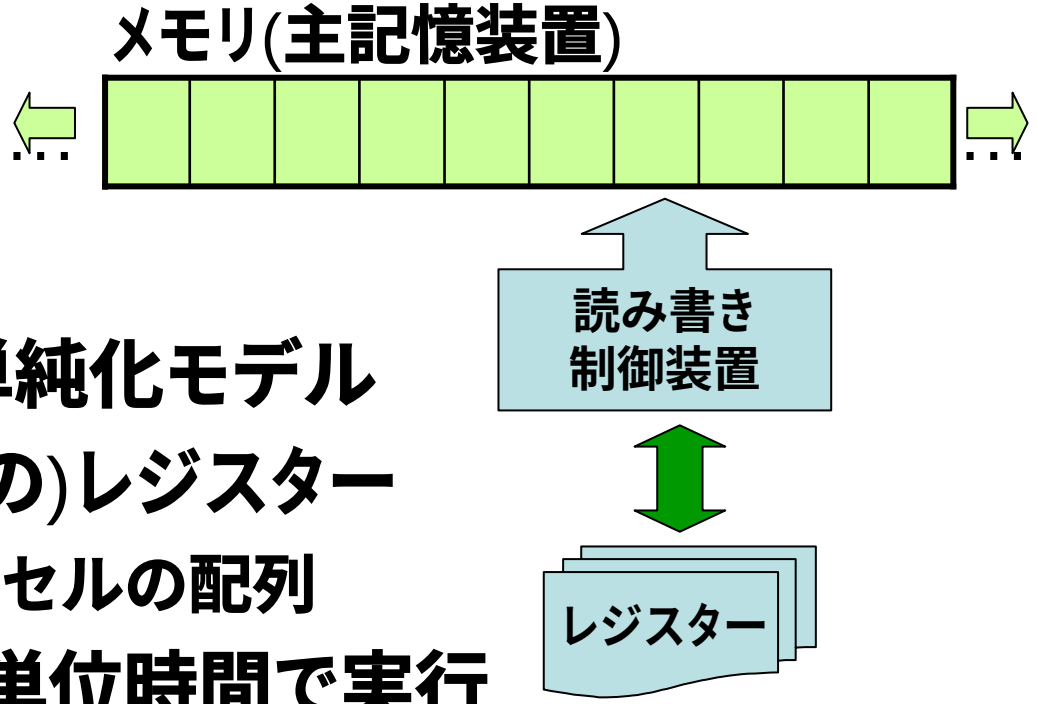
同じ土俵（モデル）が必要

計算の手間の測り方



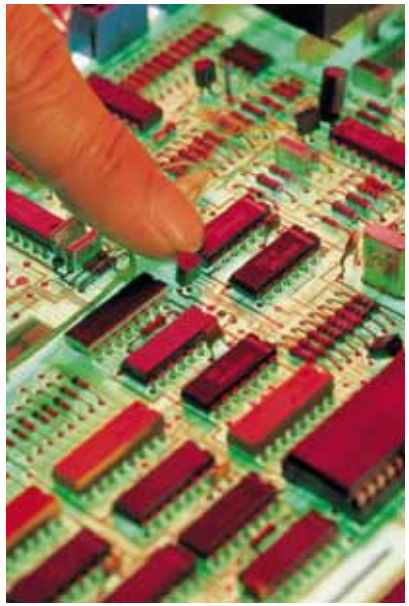
- 1回の算術演算を1単位
  何回実行したかで数える.
  - 算術演算:加減乗除・比較など
  - 足し算と掛け算の手間は一緒？
    ⇒(仮想的)コンピュータを仮定
    RAM (Random Access Machine) モデル
    ※様々なモデルが考えられる. →「計算理論」
- 入力のサイズによって手間は変化する.
  - 入力サイズに対する評価が重要

復習 RAM

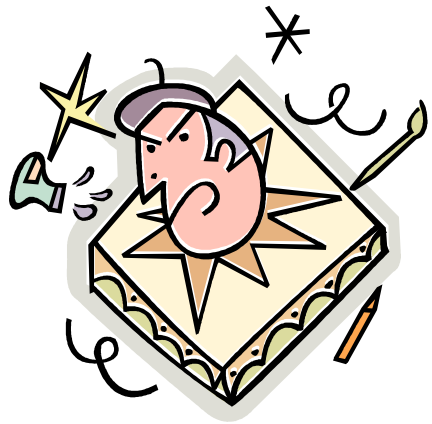


現在(将来)の計算機の単純化モデル

- (無限の)メモリと(有限の)レジスター
  - メモリ：アドレスを持ったセルの配列
- ストア，ロード，演算を単位時間で実行
  - ストア：レジスタの内容をセルに書き込む
  - ロード：セルの内容をレジスターに読み込む
  - 演算：2つのレジスタの内容を算術演算をし，結果をレジスタに書き込む



入力サイズ



- 問題を解く←問題例の入力要

入力サイズ
問題を表現するのに必要なメモリー量

- (例)
  問題：$x=\alpha$の時のn次多項式$f(x)$の値は？

$$f(x)=a_n x^n+a_{n-1}x^{n-1}+\ldots+a_1 x+a_0$$

  問題例の入力：α，nとa_n, a_{n-1}, … , a_1 , a_0

入力サイズ　$n+3$個の入力が必要

※ 入力数字が大きな時は桁数も考慮する場合がある

練習　2機械の最適加工順序問題

Q1. 8製品の場合の
入力サイズは？

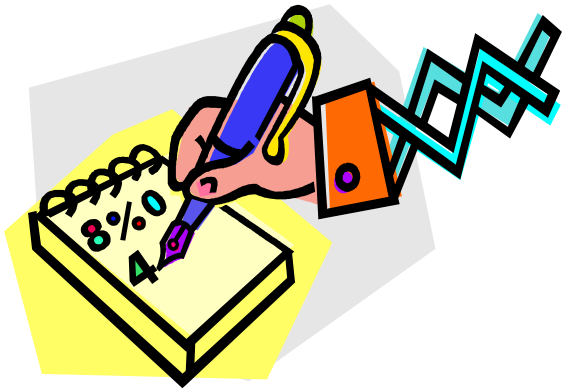
Q2. n製品の場合の
入力サイズは？

(例)

各製品の加工必要時間

8製品

製品	旋盤	研削盤
A	3	4
B	8	7
C	6	7
D	9	8
E	8	4
F	7	2
G	5	6
H	5	1



復習　ジョンソン法

2機械n製品最適加工順序問題に対する解法

1. 最小値を見つける
2. それがM1側なら…
3. 製品を消す

仮定：
今後の議論のため前処理を実施

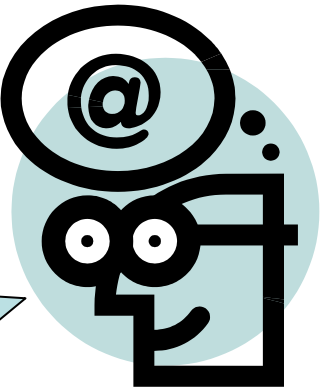
入力データ

製品	旋盤	研削盤
A	3	4
B	8	7
C	6	7
D	9	8
E	8	4
F	7	2
G	5	6
H	5	1

前処理後

min	
3	M1
7	M2
7	M1
8	M2
4	M2
2	M2
5	M1
1	M2

例題：ジョンソン法



n 製品の時，何回の比較で最適加工順序が求められる？

n回

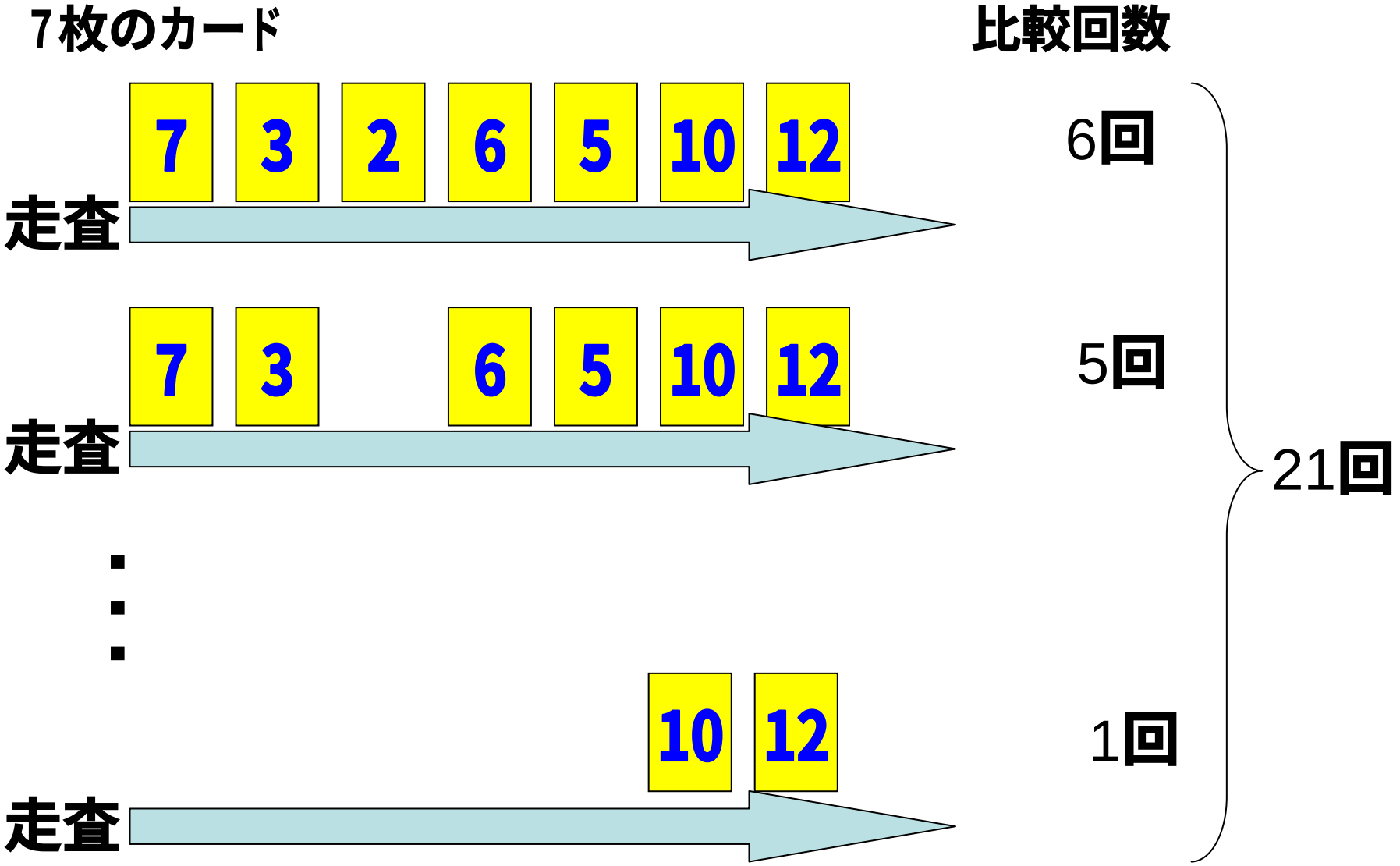
1. **最小値を見つける**
2. **それがM1側なら…**
3. **製品を消す**

最小値は何回の比較で見つかる？
↓
全部で何回？

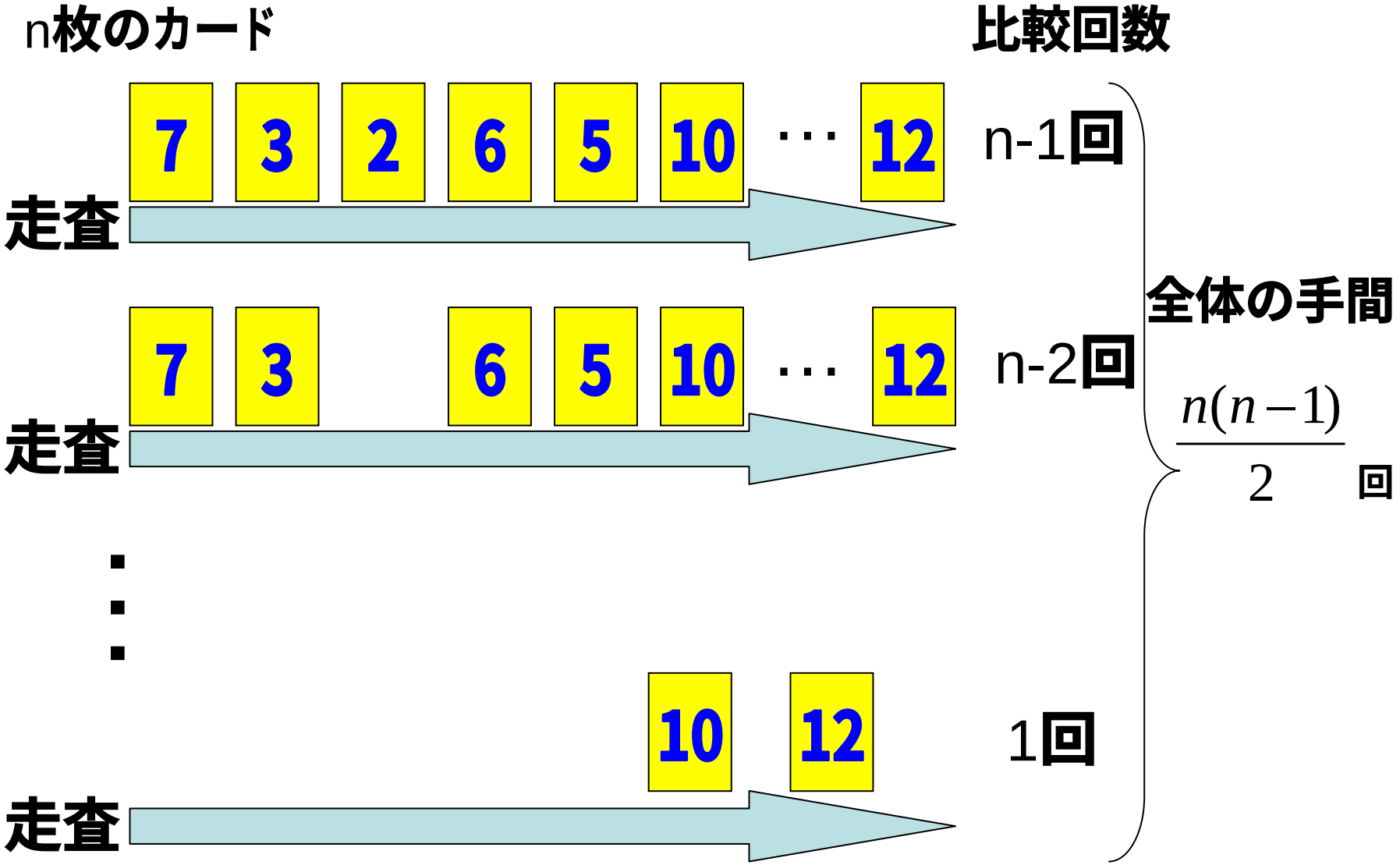
定数時間

定数時間

最小値の見つけ方 素朴な方法



素朴な方法 n枚の場合


n枚のカード
比較回数
7 3 2 6 5 10 … 12
n-1回
走査
7 3 6 5 10 … 12
n-2回
走査
10 12
1回
走査
全体の手間
$\frac{n(n-1)}{2}$ 回

工夫した方法 ヒープ

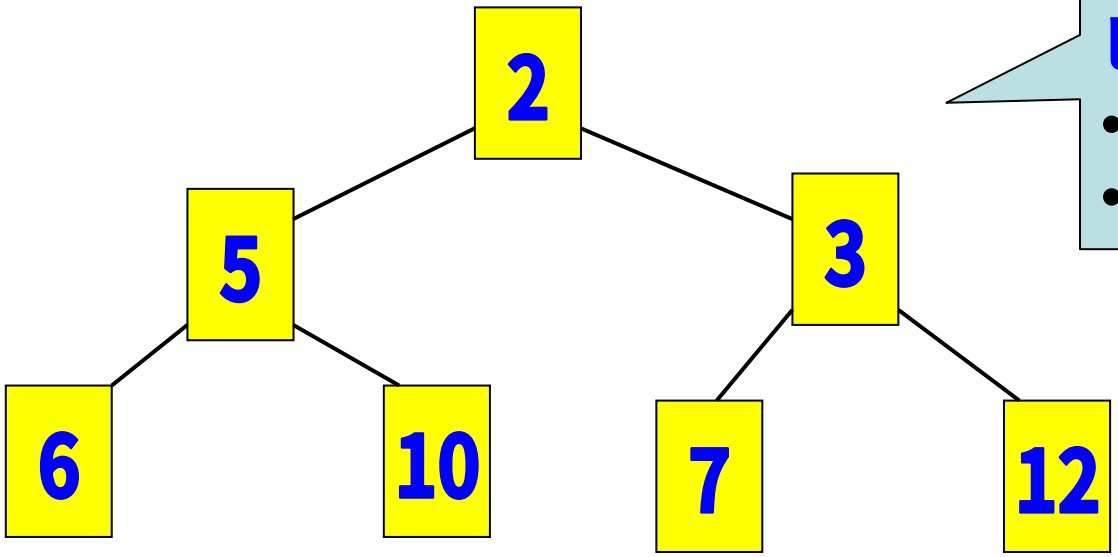
7枚のカード

7	3	2	6	5	10	12

子が2個
以下の木

ヒープ条件を
満たす
完全2分木を作成

ヒープ条件
根以外の各点で
(親の値)≦(自分の値)



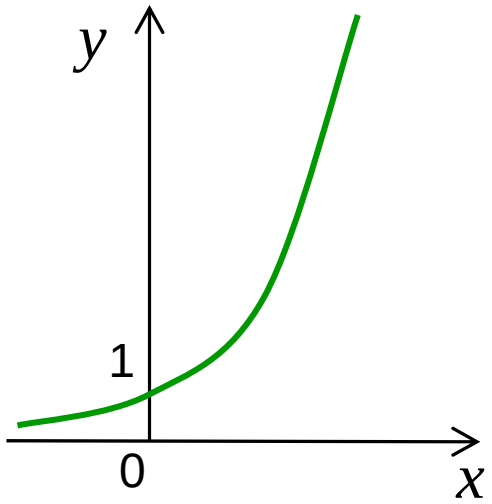
ヒープの特徴
- 根は最小値
- 木の高さは$\log_2 n$程度

対数
$2^{\alpha}=n \Leftrightarrow \log_2 n=\alpha$

復習 指数・対数

指数関数：$y=2^x$ $(=\overbrace{2\times2\times\cdots\times2}^{x\text{個}})$

x	0	1	2	3	4	5
y	1	2	4	8	16	



xが増えると，yの値が急激に増加する

対数関数：指数関数の逆関数

$y=\log_2 x$

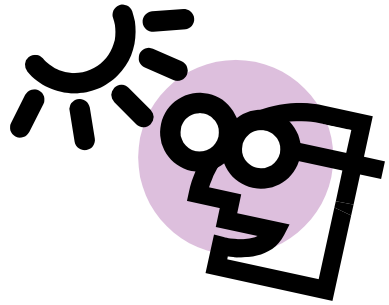
対数の底

(例) $\log_2 8=3$

$\log_2 16=4$

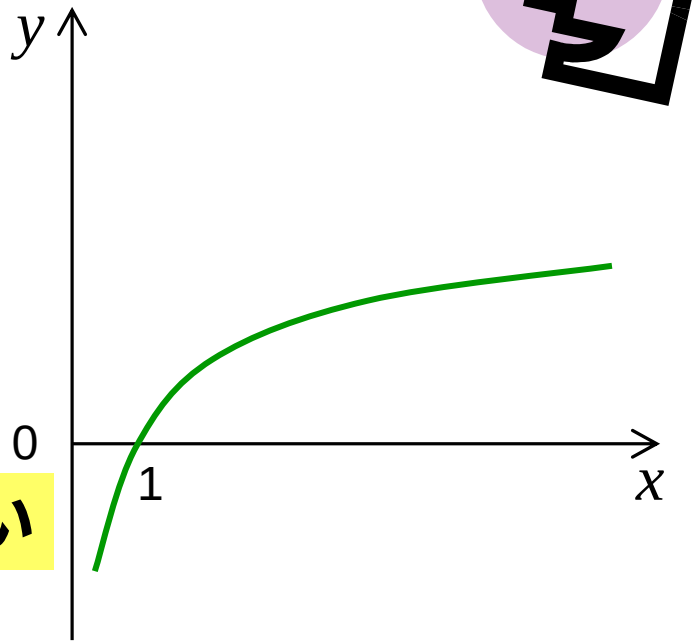
2の何乗かで8になる数は？

復習 指数･対数(2)



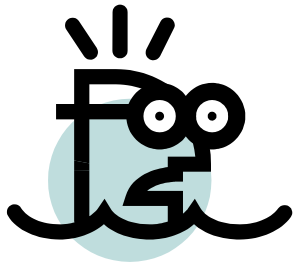
対数関数：$y=\log_2 x$

x	1	2	4	8	16	32
y	0	1	2	3	4	



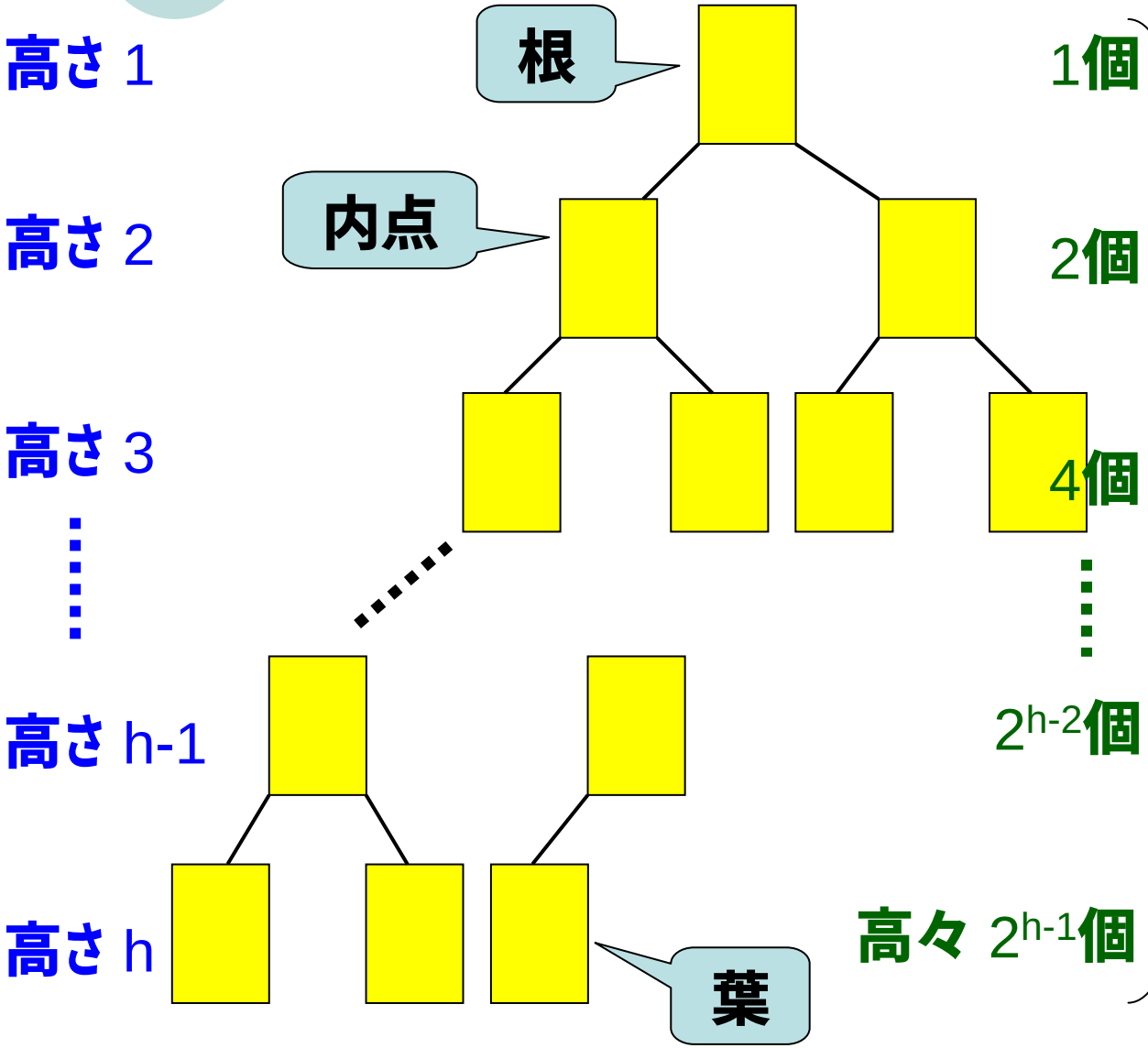
xが増加しても，yの値はあまり増えない

計算機の世界は2進数．計算機は2つの比較しかできない
→2を底にした対数がよく登場する．
(例1)アルファベットを記録するのに必要なビット数は？
(例2) 16個の金貨の中に重さの軽い偽金貨が1つある．
天秤ばかりを何回利用すると発見できる？



復習 完全2分木

- 木のグラフ
- 基本的に内点の子は2つ
- 最下層は左詰め

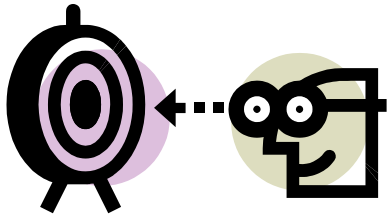


点の数がnの時の，
完全2分木の高さは？

(点の数)=
$1+2+4+\ldots+2^{h-2}+2^{h-1}$

和を求めてみよう

復習 総和を求める(2)



$1+2+4+\ldots+2^{h-2}+2^{h-1}$ の求め方

$$S=1+2+4+\ldots+2^{h-2}+2^{h-1}$$

$$-)\ 2S=\quad 2+4+8+\ \ldots\ +2^{h-1}+2^{h}$$

$$-S=1 \qquad\qquad -2^{h}$$

よって， $S=2^{h}-1$

-1は省略

$n=2^{h} \Leftrightarrow h=\log_2 n$

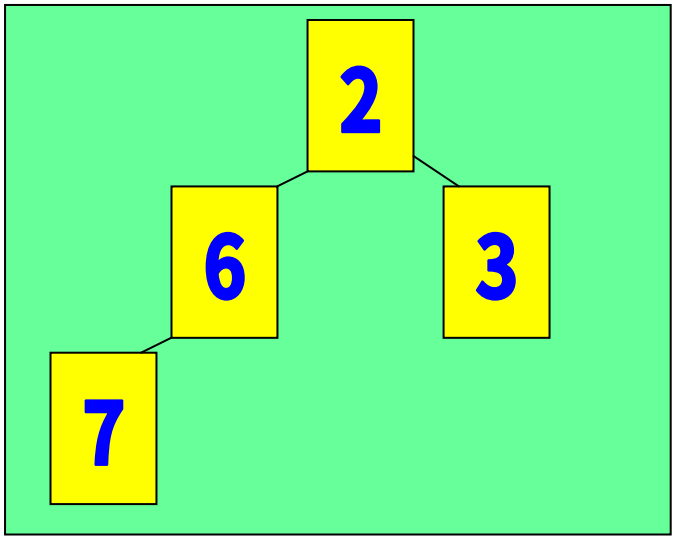
高さhの完全2分木の点の数

点数nの完全2分木の高さは$\log_2 n$

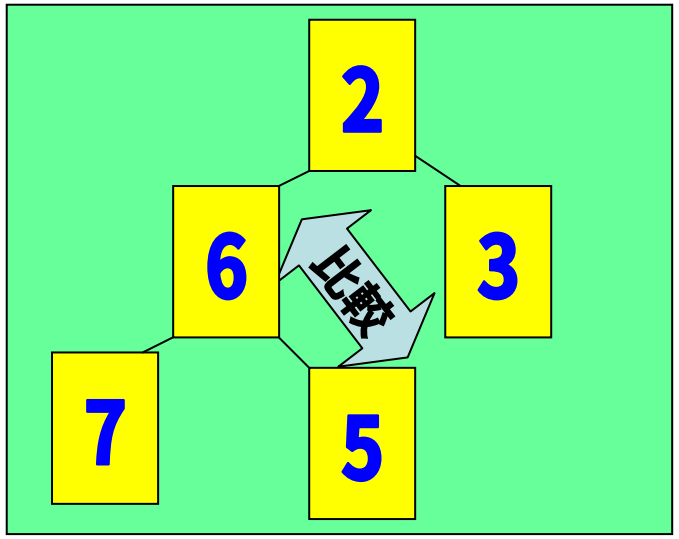
正確には，$\log_2 n$を切り上げた整数

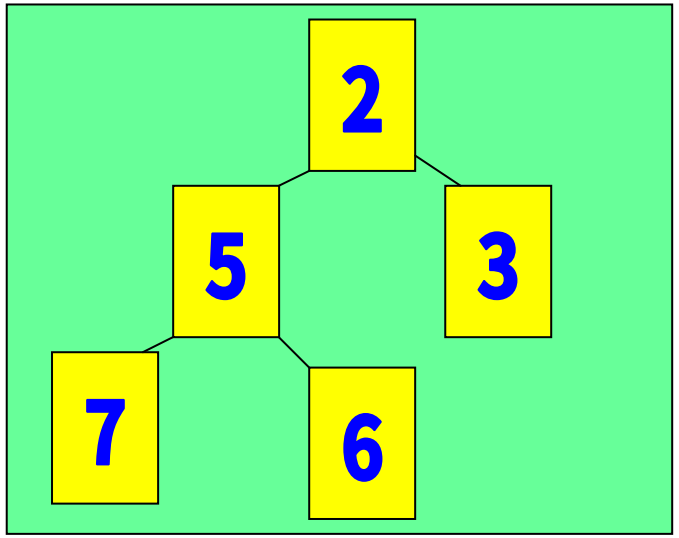
ヒープの構築方法

ヒープ



『5』を挿入





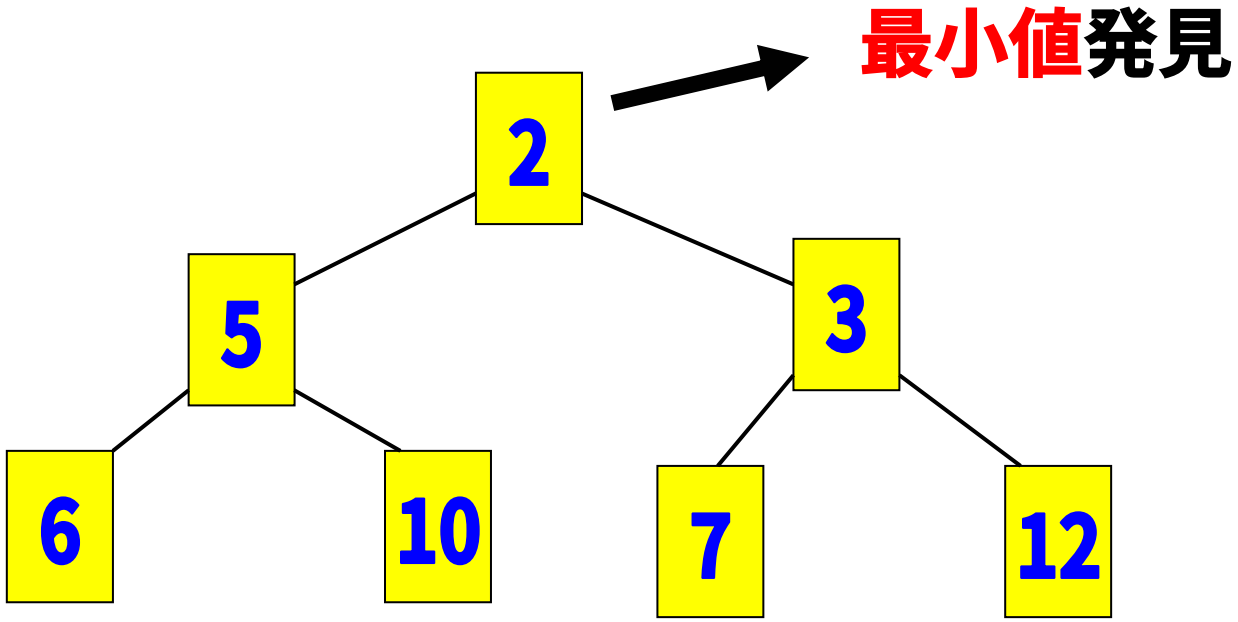
1. 最下層右に追加.
2. (繰り返し)
- 親と比較.
  自分が小さかったら交換.

1要素の挿入の手間:
最悪で木の高さ程度

ヒープの利用法

ヒープ条件より『根』の値が，
全体の最小値．

見つける手間は，
1回(定数時間)．



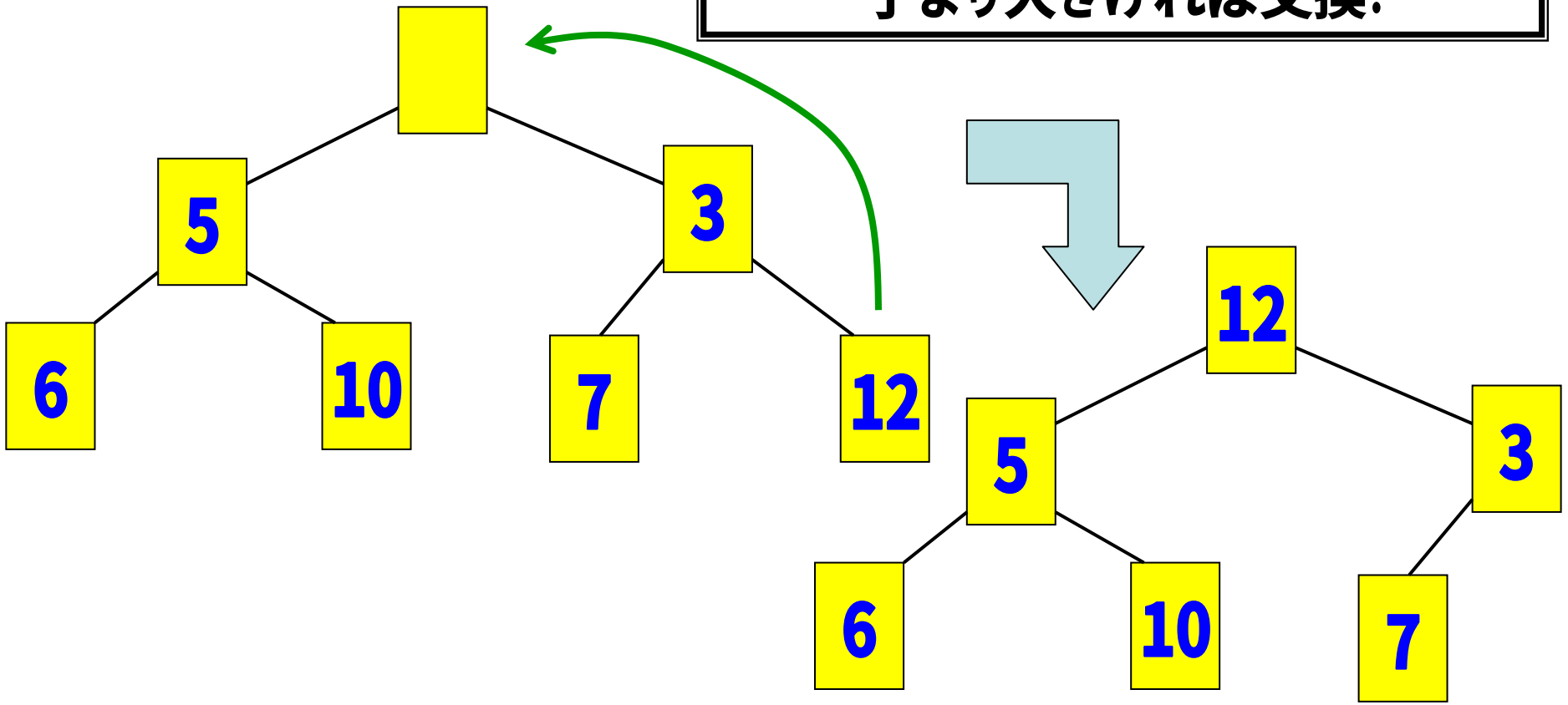
→次の最小値を見つけるには？

最小値の削除＋ヒープ条件の修復

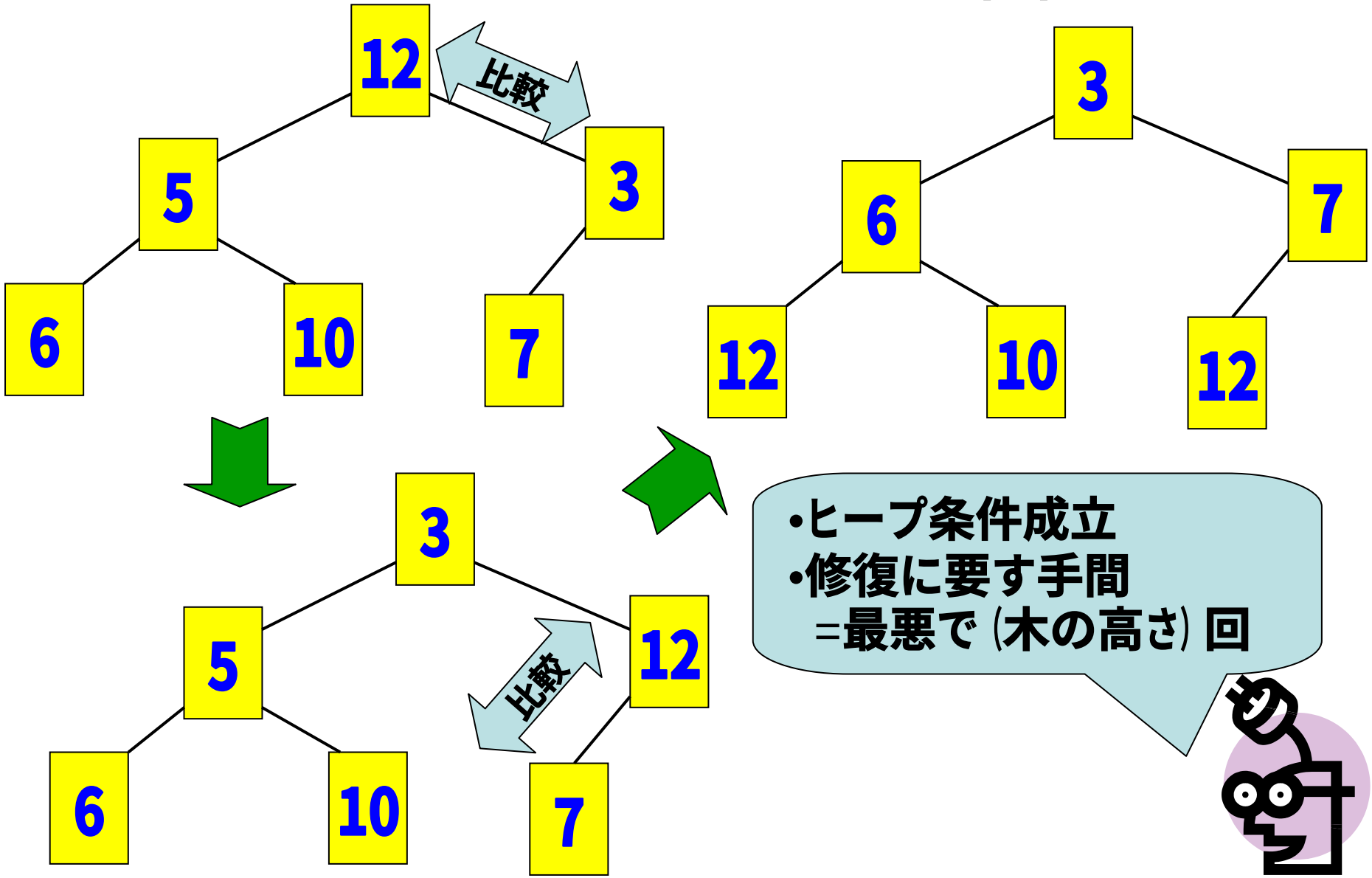
『根』を削除すると，
2分木で無い．

修復要

1. 最下層・最右の要素を根に．
2. 以下をできる限り繰り返す．
   - (2つの)子の小さい値と比較．
     子より大きければ交換．



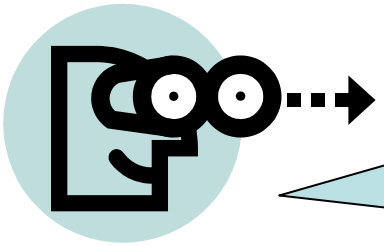
ヒープ条件の修復(2)


12
比較
3
5
6
10
7
3
6
7
12
10
12
3
5
12
比較
6
10
7
•ヒープ条件成立
•修復に要す手間
=最悪で(木の高さ)回

ヒープ利用時の手間

n個の要素から最小値を順に見つける手間

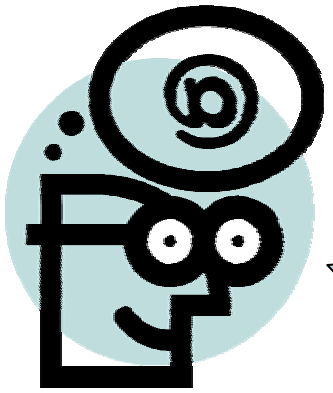
- (準備)ヒープの構築：
  (1要素挿入手間)×(要素数)＝最悪で(木の高さ)×n
- 最小値の発見：n回
  - 1回の最小値の発見（＝親の値）：1回
  - 1回のヒープ修復の手間：最悪で（木の高さ）



完全2分木の高さ
＝$\log_2 n$

確認してみよう

最悪の場合，全体で
$2n\log_2 n$回程度の手間



例題(続) ジョンソン法

製品数がnの時，
計算の手間は？

準備が必要
$n\log_2 n$

n**回**

1. **最小値を見つける**
2. **それがM1側なら…**
3. **製品を消す**

ヒープを利用：最悪 $\log_2 n$
素朴な方法：最悪 n

定数時間

定数時間

⇒ 最小値発見が計算の手間に決定的

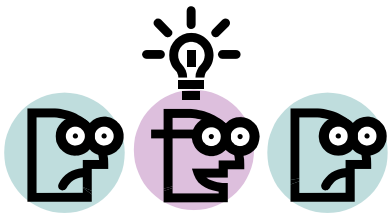
アルゴリズムの比較

n個の要素から最小値を順に取り出す手間は？

素朴な方法
- ちょうど $n(n-1)/2$ 回

ヒープを利用した方法
- 実際にどの程度かかるかは不明
- 最悪で $2n\log_2 n$ 回

1. n=8の時,各々の方法は何回の演算を実行している？
2. n=1,000,000の時, 各々の方法は約何回の演算を実行する？
3. 1億回/秒演算できるコンピューターがある.
   上記2のとき, 各方法では何秒で作業を終了するか？



最悪の状況で比較することに意味がありそうだね.

最悪計算(時間)量

同じサイズの入力に対して，

最悪時のアルゴリズムの手間を測定した計算量

⇒アルゴリズムを評価する物差しにする．

アルゴリズムを評価する
他の主な物差し

- 平均計算量
- 最悪計算領域量：アルゴリズム実行に使用するメモリーの量を評価する．

等

漸近計算量

- **計算量は簡潔な表現を用いることが多い**
  - n→∞になるときの挙動が知りたい
  - **(例)ある最悪計算量**：$2n^2+5n+120$
    nが大きくなったとき計算量に支配的に影響するのはn^2の項のみ．
    →ほかの情報を省いても，挙動は把握可能
- **簡潔な表現方法：最も影響する項だけで表現**
  - $2n^2+5n+120 \Rightarrow O(n^2)$

どうしてだろう？
考えてみよう！

「オーダー」と読む
アルファベットの大文字『O』

ある正定数cとNが存在し，N以上のnに対しT(n) ≦ cf(n)が成立
⇔
T(n)＝O (f(n))

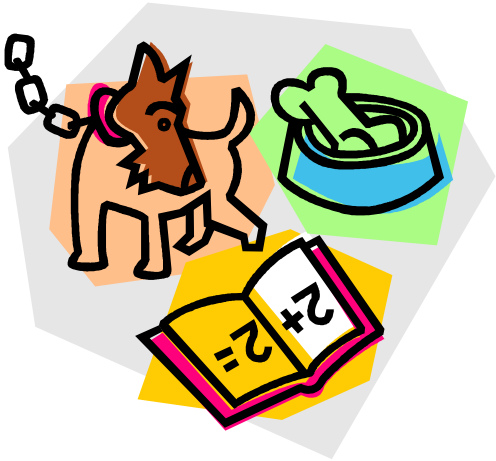
$O(f(n))$

厳密には，$O(n^3)$でもよいので嘘．ただ，実用上はこの理解でも良い．詳しい内容は別な機会に．

- $2n^2+3n+10= O(n^2)$
- $100000n^2-100n= O(n^2)$
- $0.00001n^2+10000000= O(n^2)$

どれも漸近計算量で表現すると同じ

- $O(1)$：**定数時間オーダー**
  - **入力サイズに依存しないことを示す**



世の中の約束事

O(オーダー)の記法を「＝」の式で用いるときは右辺に書く．
左辺は右辺より低い情報量になることは無い．

※オーダーの記法は情報を省略している

例題

n個の要素から最小値を順に取り出す手間は？

素朴な方法

- **ちょうど $n(n-1)/2$ 回**
- **最悪計算量 $n(n-1)/2$**

$O(n^2)$

ヒープを利用した方法

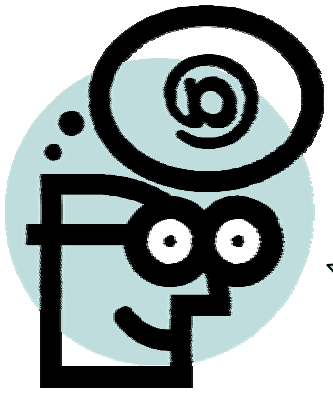
- **最悪で $2n\log_2 n$ 回**
- **最悪計算量 $2n\log_2 n$**

$O(n\log_2 n)$

大きなnに対して，$n>\log n$ ⇒ $n^2>n\log n$

計算量の意味で

⇒ ヒープ利用の方法は素朴な方法より優れている



例題(続) ジョンソン法

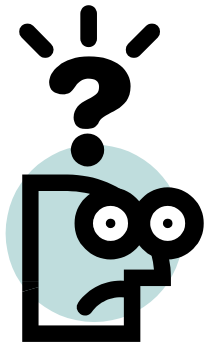
製品数がnの時，
計算の手間は？

準備が必要
$O(n\log_2 n)$

n回

1. **最小値を見つける** — **ヒープを利用**：$O(\log_2 n)$
2. **それがM1側なら…** — **定数時間**
3. **製品を消す** — **定数時間**

⇒ ヒープを利用して $O(n\log_2 n)$



もっと速くなる？

ジョンソン法の改造

入力データ

製品	旋盤	研削盤
A	3	4
B	8	7
C	6	7
D	9	8
E	8	4
F	7	2
G	5	6
H	5	1

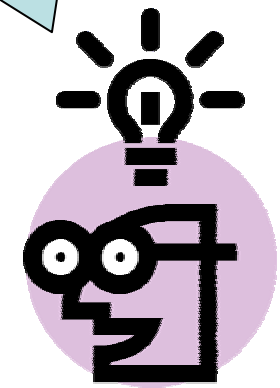
前処理

min	
3	M1
7	M2
7	M1
8	M2
4	M2
2	M2
5	M1
1	M2

さらに前処理
min順で並び替え

製品	min	
H	1	M2
F	2	M2
A	3	M1
E	4	M2
G	5	M1
B	7	M2
C	7	M1
D	8	M2

この表を使えば，あとは簡単！

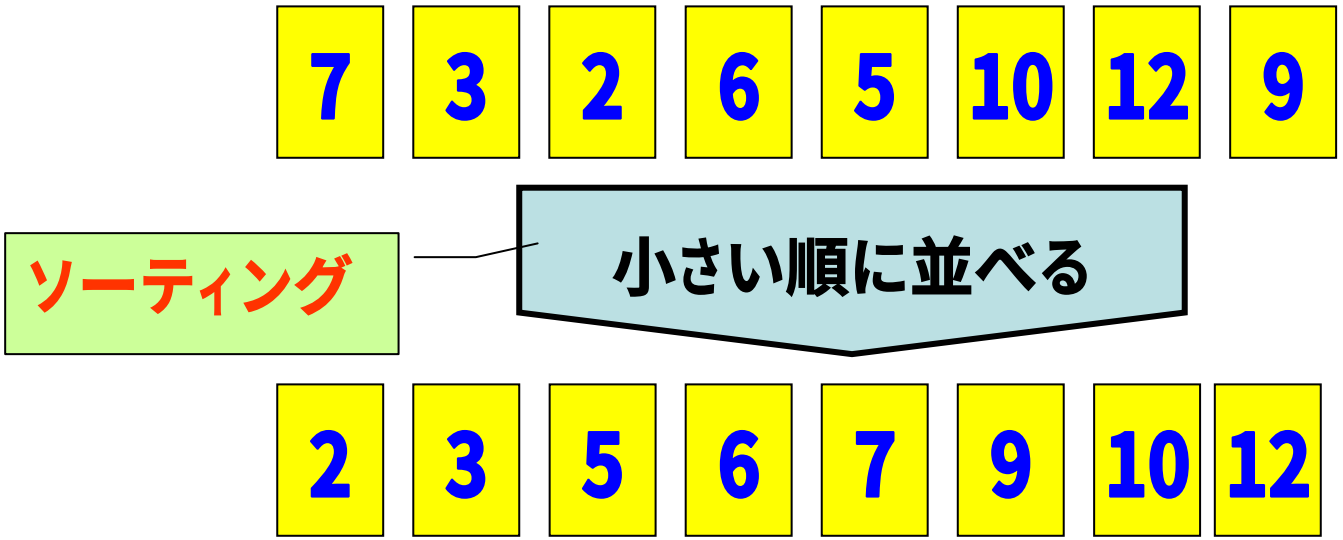


ソーティング

並び替えの手間は？

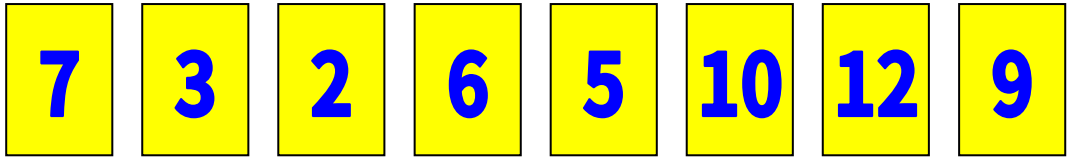
ソーティング

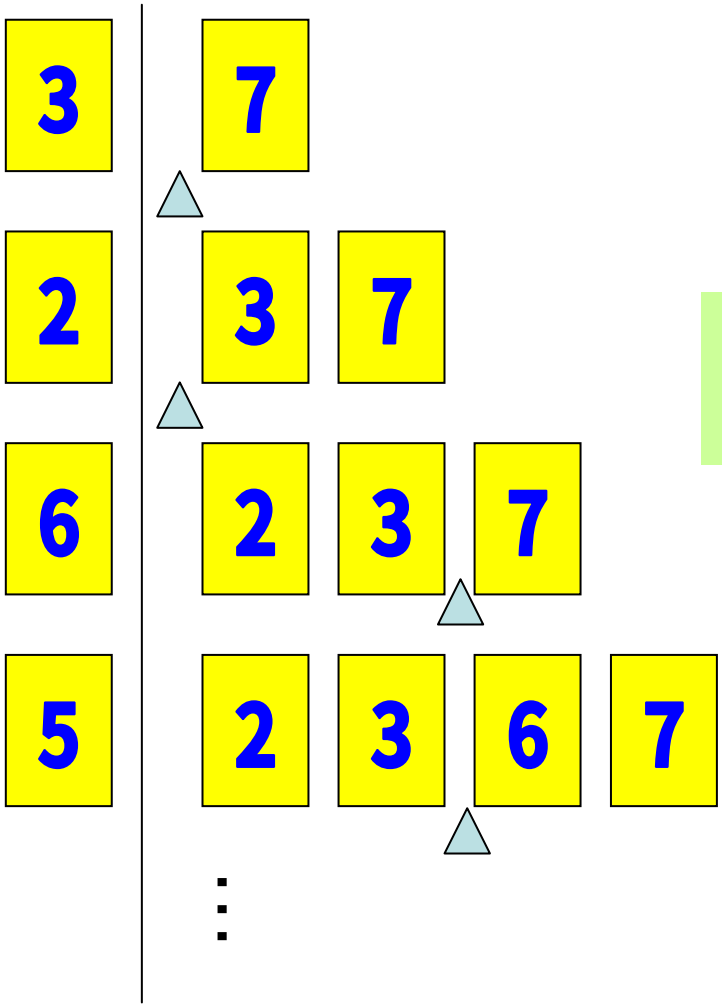
8枚のカード



ソーティング 挿入法

8枚のカード





比較回数は
最悪何回？

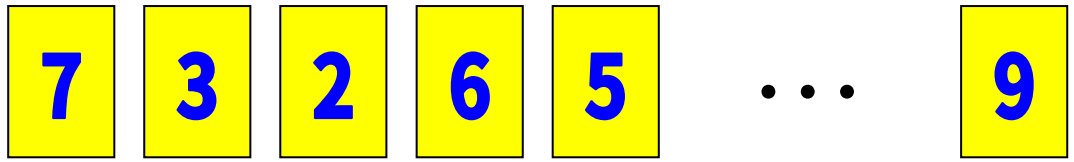
最悪比較回数

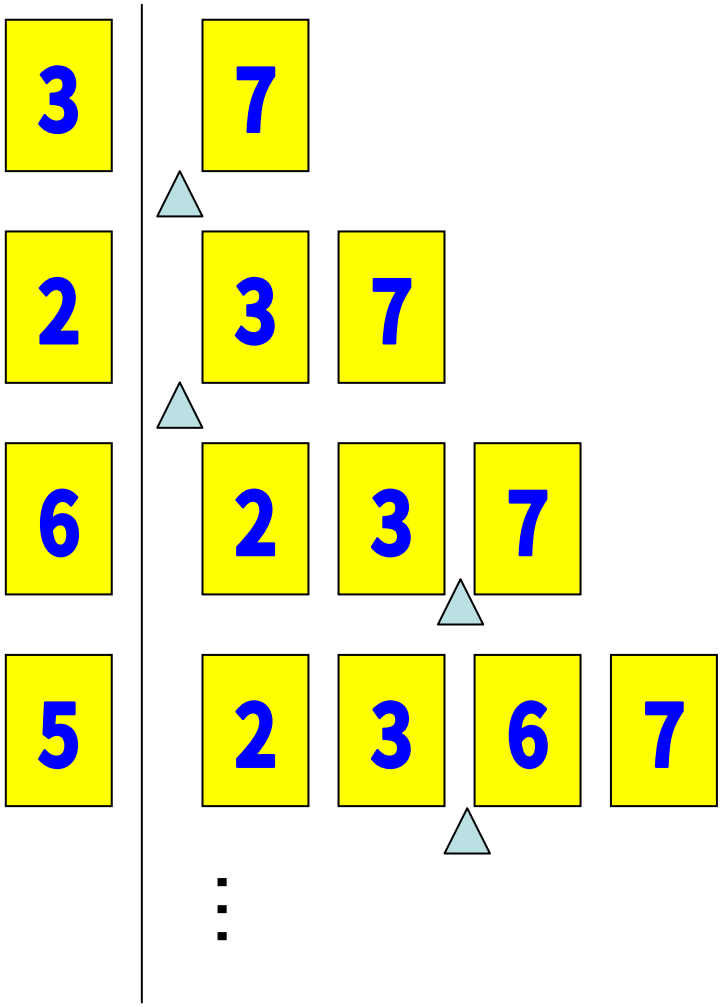
1回
2回
3回
4回
⋮
7回

28回

ソーティング 挿入法 n枚の場合

n枚のカード





最悪比較回数

1回

2回

3回

4回

⋮

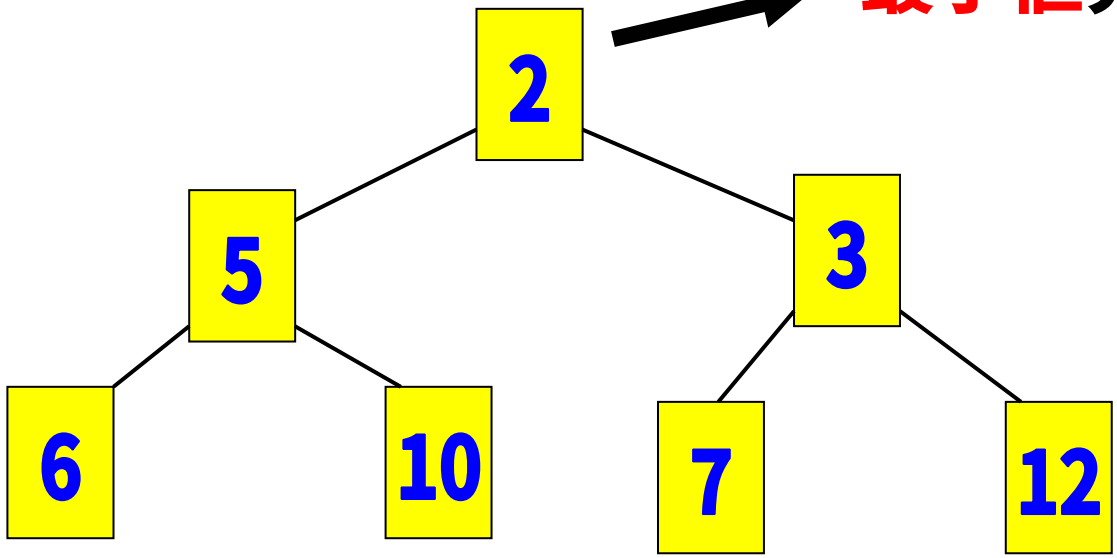
n回

$\frac{n(n+1)}{2}$ 回

$O(n^2)$

ヒープの利用したソーティング

ヒープ条件より『根』の値が，
全体の最小値．



O(1)

最小値発見

O(log$_2$n)

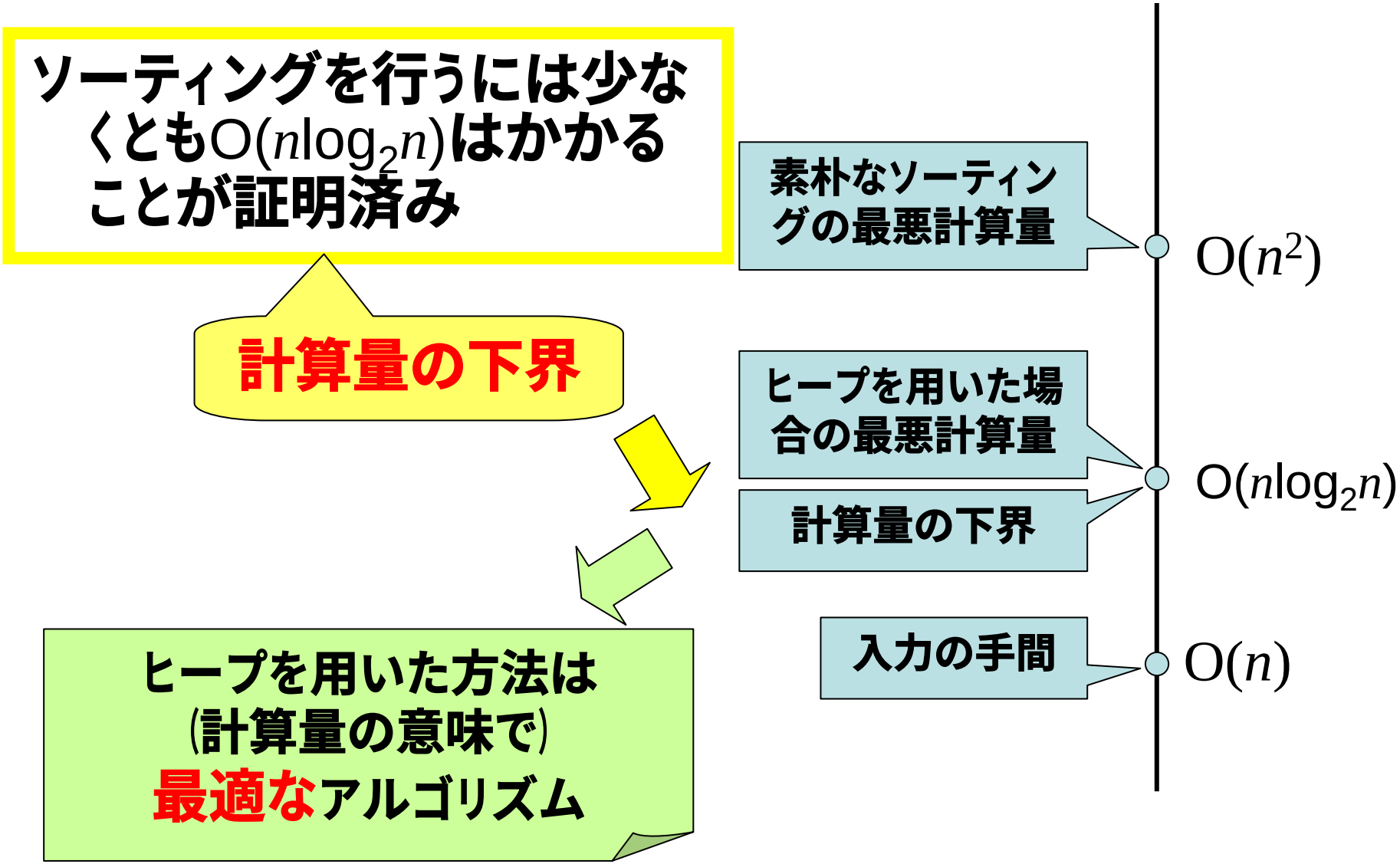
ヒープの修正

n回繰り返し

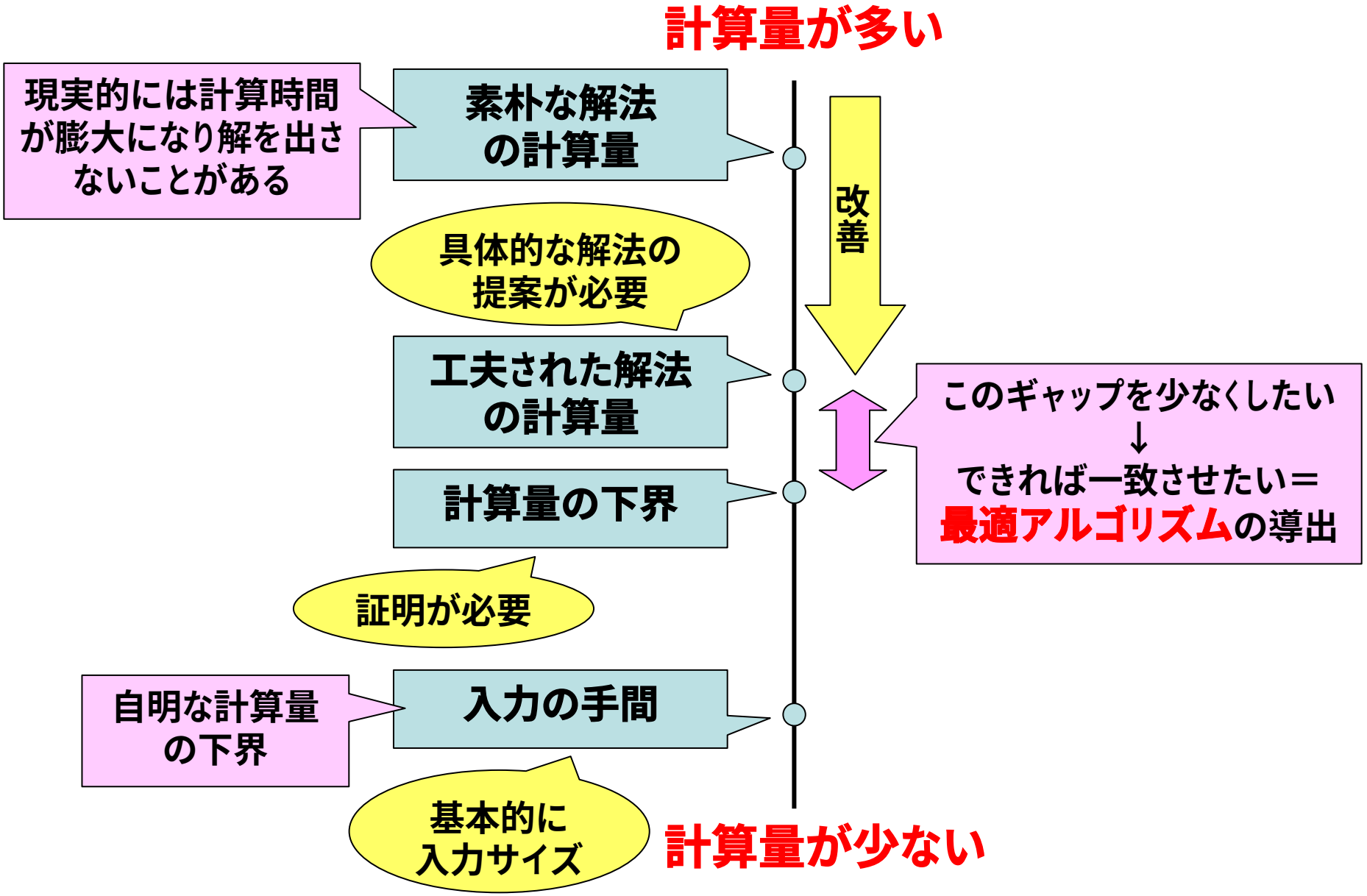
O(nlog$_2$n)

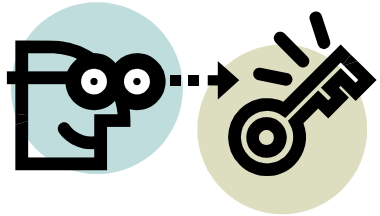
ヒープソート

ソーティングの計算量の下界



ある問題の計算量の構造





ここまでのまとめ

- ある問題に対するアルゴリズムは複数
- アルゴリズムは最悪計算量で評価
- 最悪計算量は漸近計算量で表すと便利
  ⇒計算量の意味で優劣比較が可能

•計算量のみが優劣の基準ではない
•状況に応じた別尺度の導入も大切
•ただし，最適化問題では重要要素

next

問題自体が持つやさしさ･難しさ

演習2-2 最小木問題

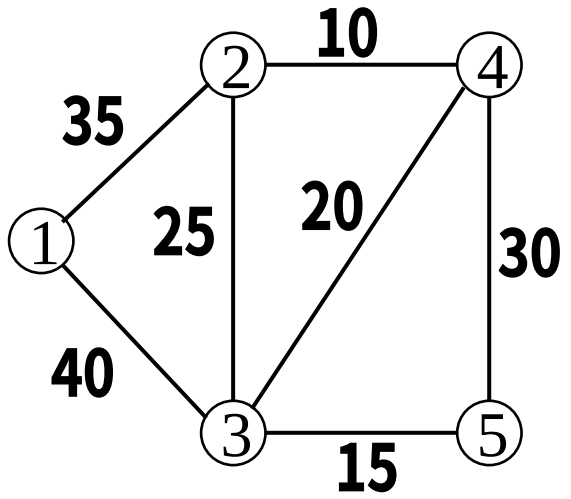
点数がn個，枝数がm本のグラフの重み和最小の木を探せ

Kruskalの解法

1. T=φ
2. 枝の重みの小さい順に以下の (*) を行う.

   (*) Tに枝を加えた時に，Tに閉路ができなければその枝をTに加える.

最終的に得られたTが最小木



① 問題の入力サイズは？
② Kruskalの解法を効率よく実行する方法は？
③ その最悪計算量は？