プログラミング発展

2023年度2Q　火曜日5~7時限(13:45~16:30)
金曜日5~7時限(13:45~16:30)

工学院　情報通信系

尾形わかは，松本隆太郎，
Chu Van Thiem，Saetia Supat
TA:東海林郷志，千脇彰悟

（最終更新：６月13日１６時）

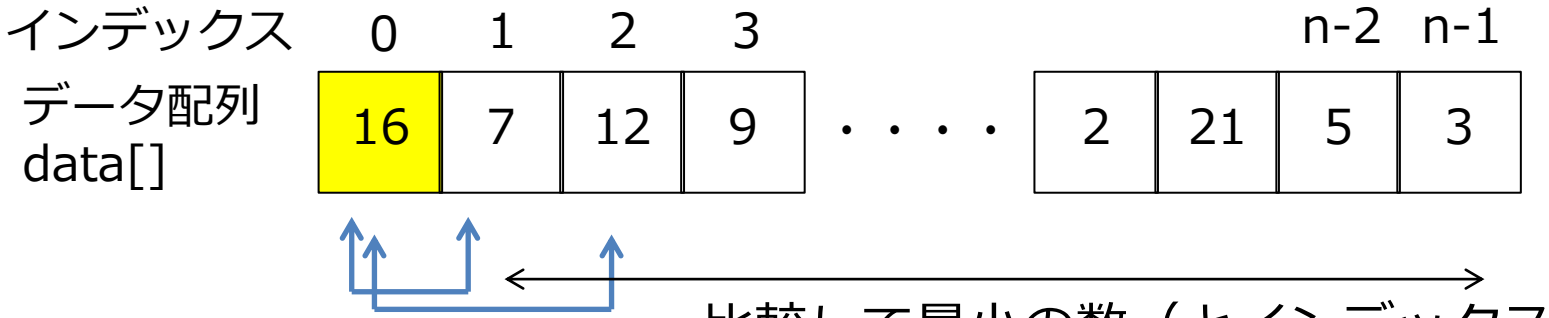
「アルゴリズム的考え方の基礎２」

1. プログラミング基礎の課題プログラムとアルゴリズム的考え方の基礎
2. 文字と文字列の復習の課題

アルゴリズム的考え方の基礎2

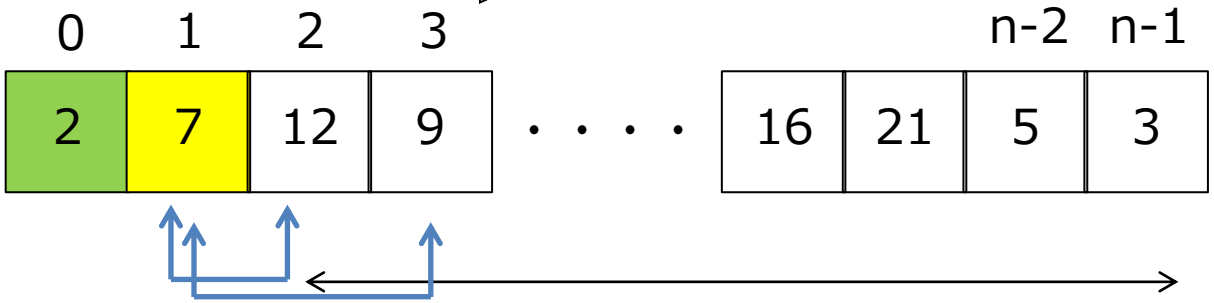
・ソーティング

- 単純なソーティング（選択ソート）



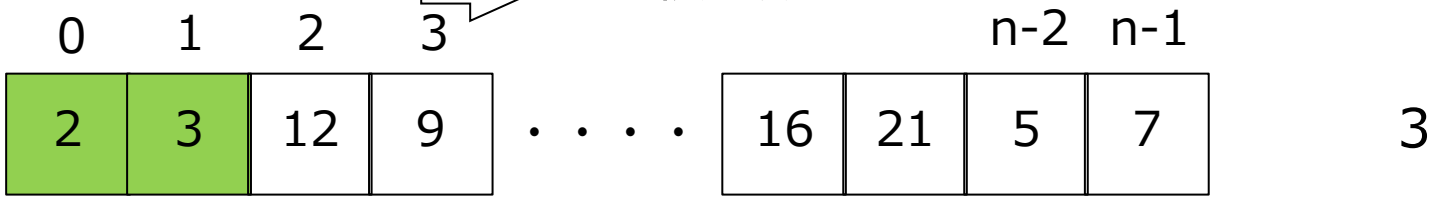
比較して最小の数（とインデックス）を求める

⇨ 比較回数 n-1



比較して最小の数（とインデックス）を求める

⇨ 比較回数 n-2



アルゴリズム的考え方の基礎2

・ソーティング

- 単純なソーティング（選択ソート）

最外ループ（繰り返し）で

比較回数 n-1, n-2, ‥ ‥

入替回数　1,　 1,　　‥ ‥

⇨　全体の比較回数で計算量を評価

総比較回数= (n-1) + (n-2) + ‥ ‥ + 1 = $\frac{n\,(n\text{-}1)}{2}$

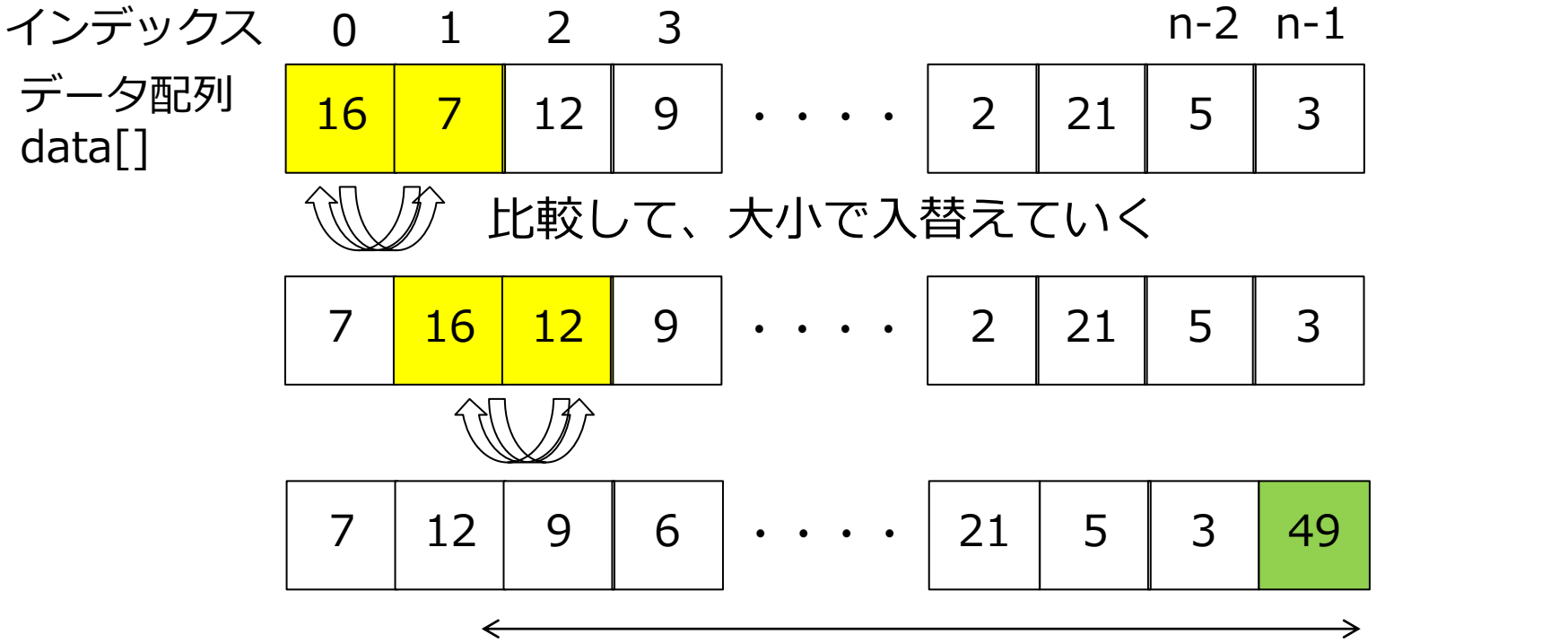
⇨総比較回数はn^2に比例→実行時間もn^2に比例

“O(n^2)”（オーダーn^2）(漸近的時間計算量)と表記します

アルゴリズム的考え方の基礎2

・ソーティング

- 単純なソーティング（交換法による（バブル）ソート）



比較＆入替回数 n-1

＊ 比較はn-1，入替はそれ以下

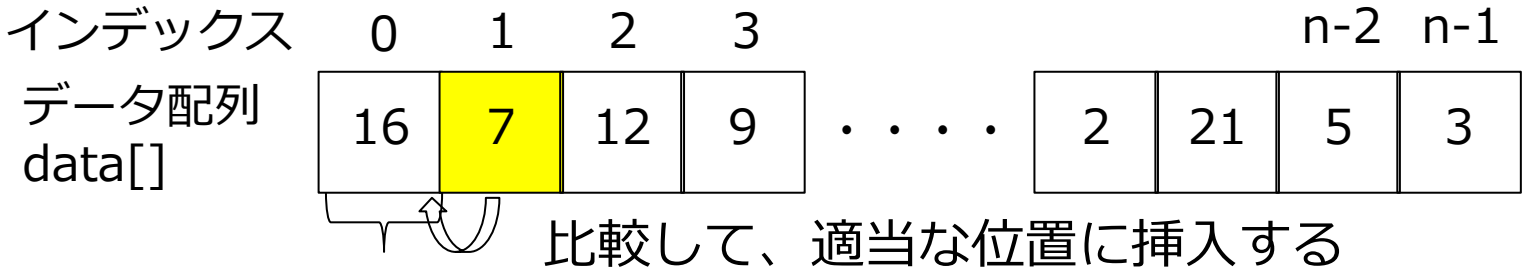
残り(緑以外)の部分について同様の操作：比較＆入替回数 n-2, ・・・・, 2, 1

⇨総比較操作回数（比較＆入替）：$O(n^2)$

アルゴリズム的考え方の基礎2

・ソーティング

- 単純なソーティング（挿入法によるソート）



比較回数1

比較回数2以下

:

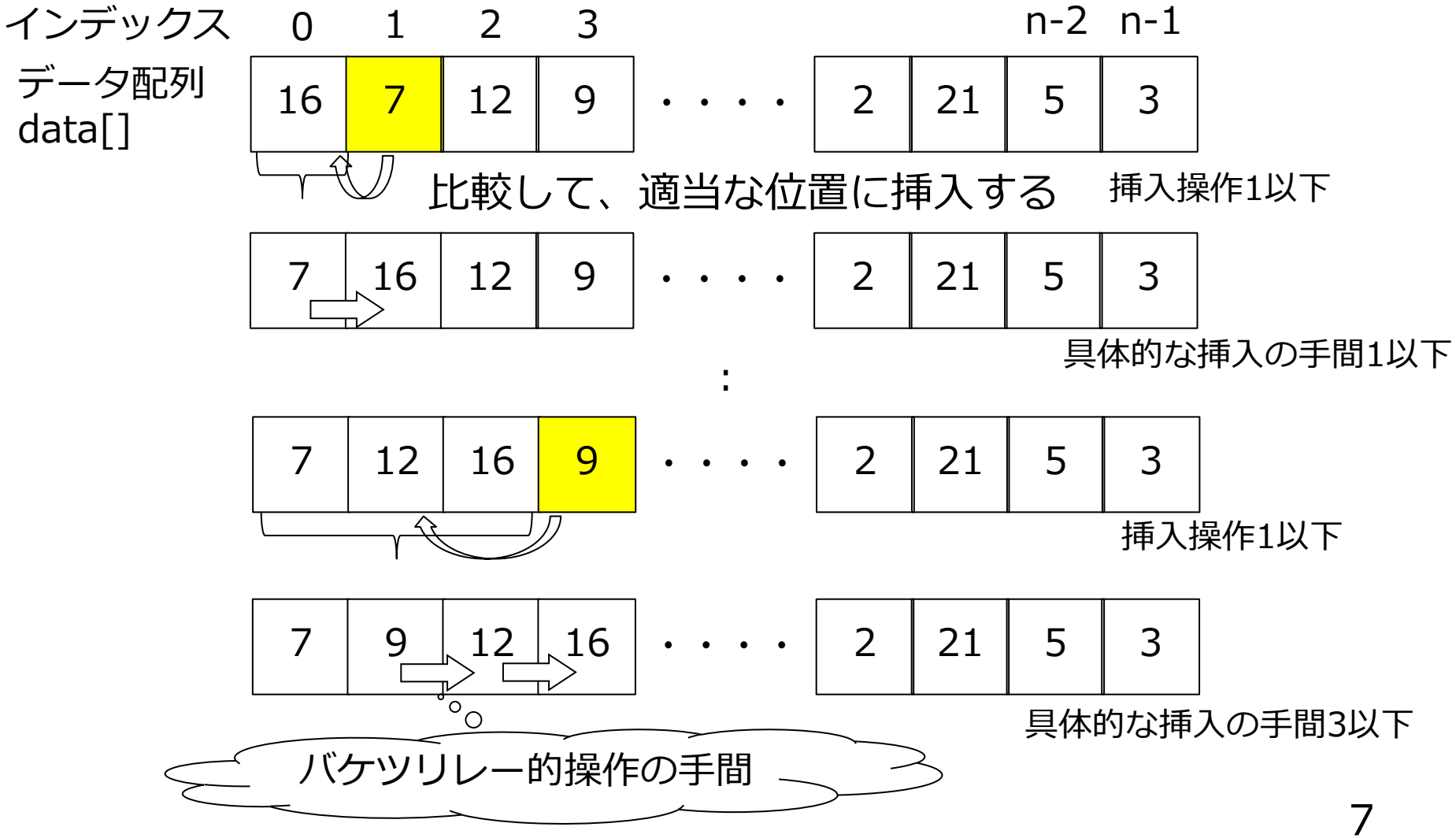
⇨ 最悪の総比較回数=1+2+ ‥ ‥ + (n-1) = $\frac{n\,(n\text{-}1)}{2}$

⇨ 最悪総比較回数: O(n^2)

アルゴリズム的考え方の基礎2

・ソーティング

- 単純なソーティング（挿入法によるソート）



アルゴリズム的考え方の基礎2

・ソーティング

- 単純なソーティング（挿入法によるソート）

⇨ 最悪の挿入操作 = n

最悪の挿入の手間=$1+2+\cdots\cdots+(n-1)=\frac{n(n-1)}{2}$

⇨ 最悪の挿入手間: $O(n^2)$

最悪となる例

⇨ 挿入の手間（$O(n^2)$）$\begin{matrix}>\\<\\?\end{matrix}$ 比較の手間 （$O(n^2)$）

アルゴリズム的考え方の基礎2

・ソーティング

- 単純なソーティング（まとめ）

選択ソート

最悪／平均比較回数 $O(n^2)$

交換法によるソート（バブルソート）

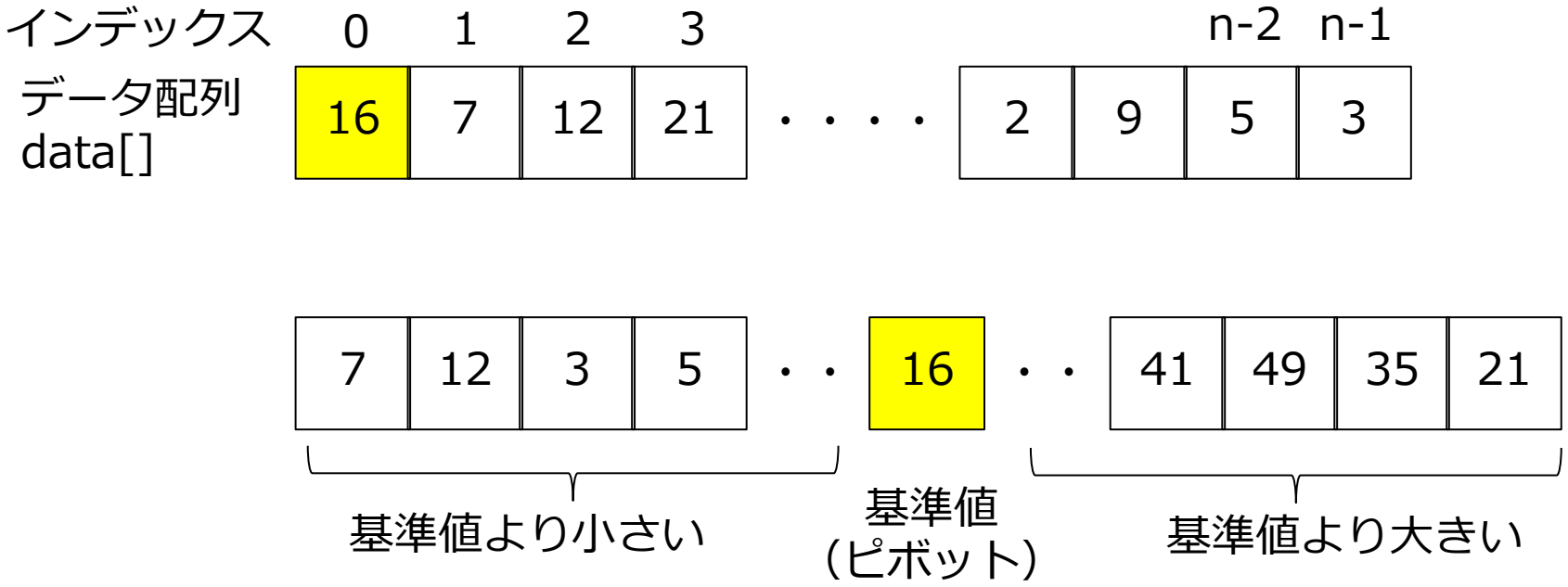
比較＆入替回数 $O(n^2)$

挿入法によるソート

比較回数（最悪） $O(n^2)$

アルゴリズム的考え方の基礎2

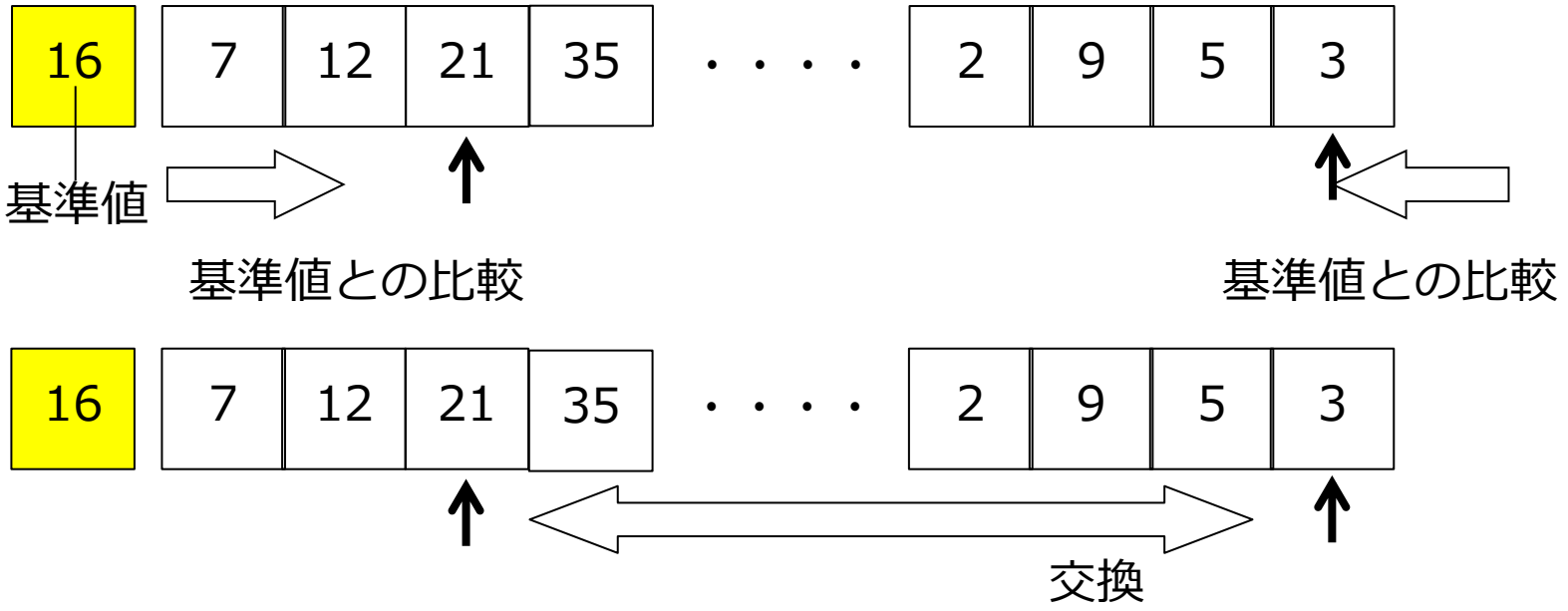
・ソーティング
　- クィックソート



アルゴリズム的考え方の基礎2

・ソーティング
　- クィックソート

データを２つに分ける操作(partition)の計算量（計算時間）

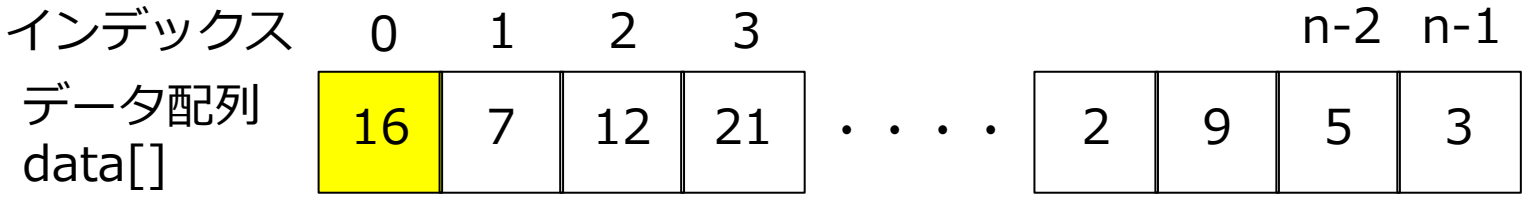


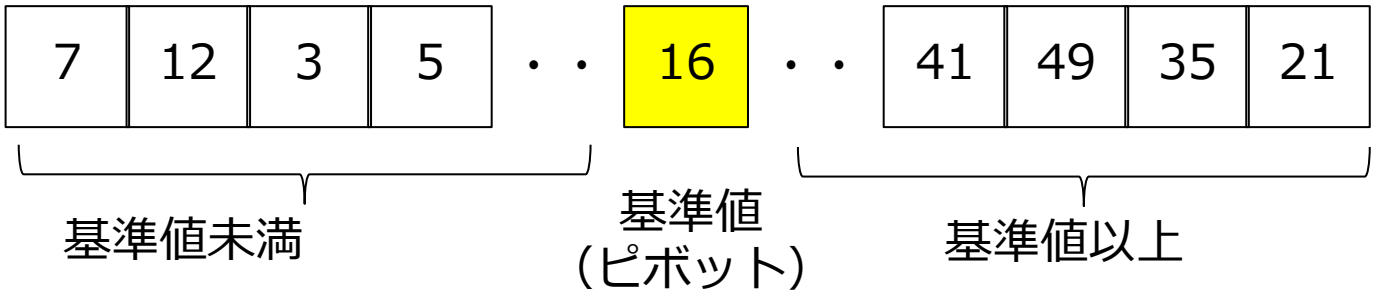
比較回数：n　　比較回数 O(n)

入替回数：n/2以下　　入替回数（最悪） O(n)

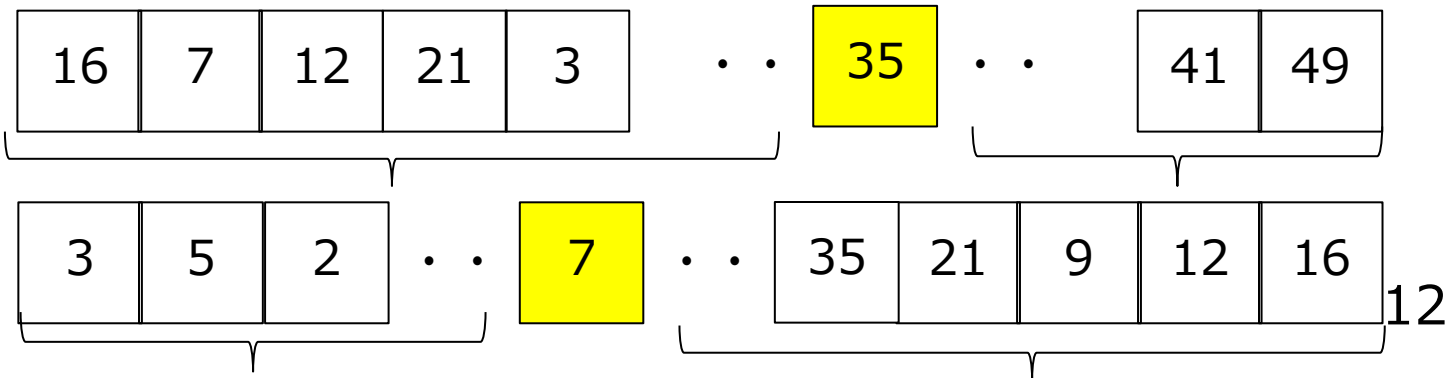
アルゴリズム的考え方の基礎2

・ソーティング
　- クィックソート



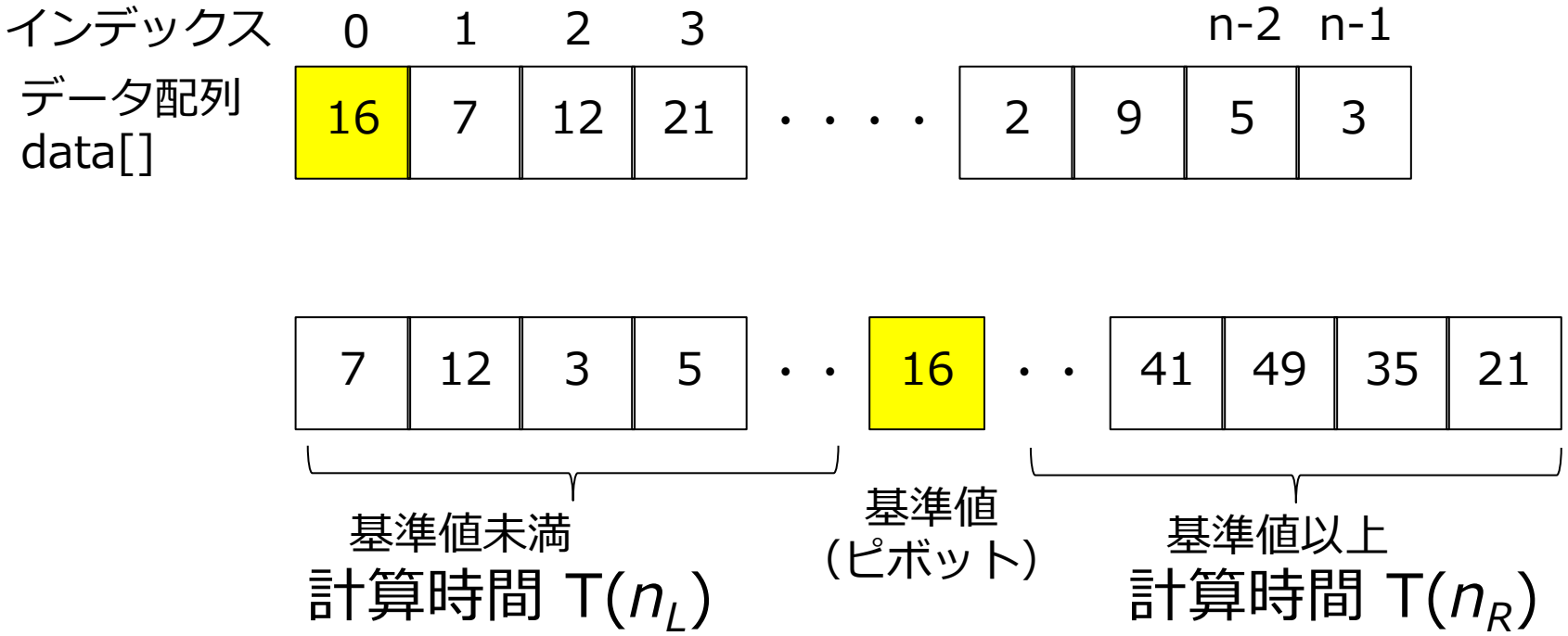


全体の計算量➔基準値の選び方による



アルゴリズム的考え方の基礎2

・ソーティング
　- クィックソート



全体の計算時間T(n)=partitionの計算時間+T(n_L)+T(n_R)

T(n_L) = partitionの計算時間+T(n_{LL})+T(n_{LR})

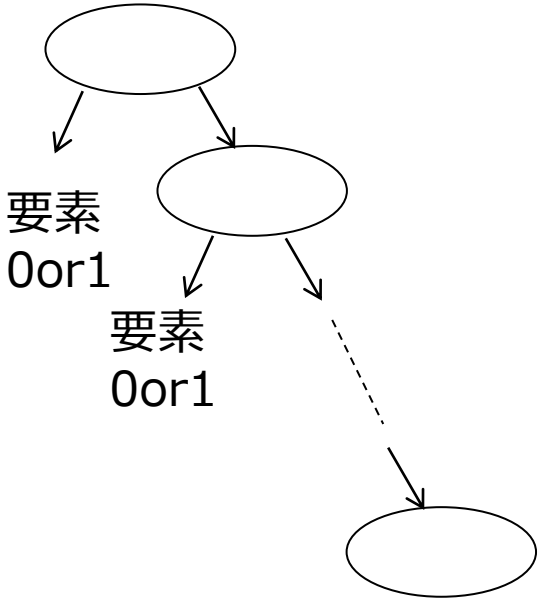
T(n_R) = partitionの計算時間+T(n_{RL})+T(n_{RR})

全体の計算量➔各partitionでの基準値の選び方による

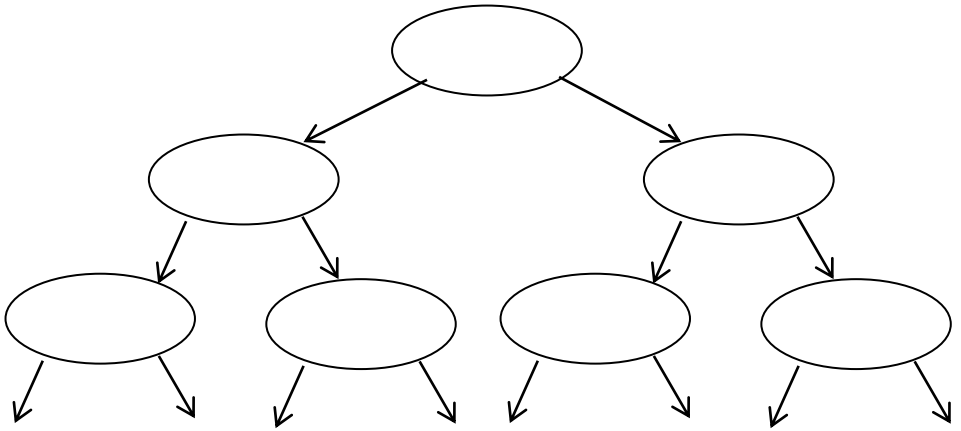
アルゴリズム的考え方の基礎2

・ソーティング
 - クィックソート

全体の計算時間T(n)=partitionの計算時間+T(n_L)+T(n_R)
T(n_L) = partitionの計算時間+T(n_{LL})+T(n_{LR})
T(n_R) = partitionの計算時間+T(n_{RL})+T(n_{RR})
全体の計算量➔各partitionでの基準値の選び方による



最悪の場合
（極端に偏った分け方の場合）

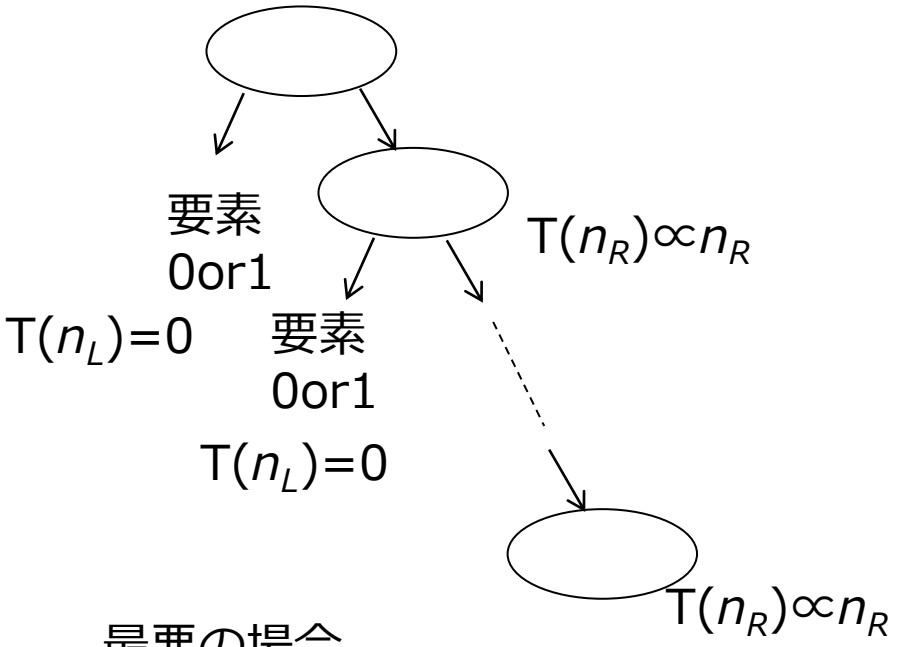


最良の場合
（非常にバランスのとれた分け方の場合）

アルゴリズム的考え方の基礎2

・ソーティング
- クィックソート

全体の計算時間T(n)=partitionの計算時間+T(n_L)+T(n_R)
T(n_L) = partitionの計算時間+T(n_{LL})+T(n_{LR})
T(n_R) = partitionの計算時間+T(n_{RL})+T(n_{RR})
全体の計算量➔各partitionでの基準値の選び方による



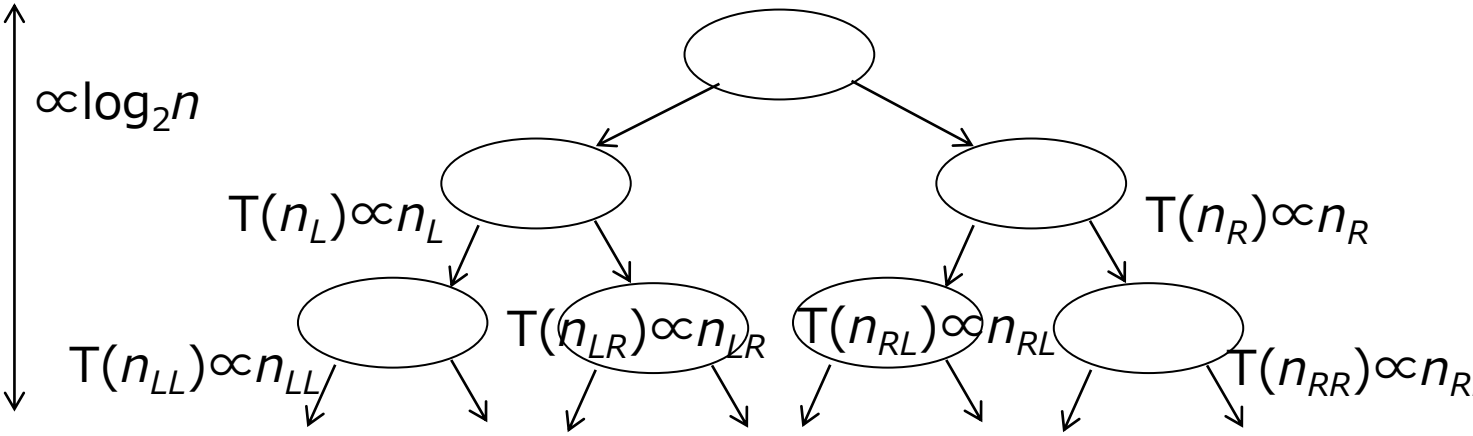
partitionの計算時間（最悪）
O(n^2)

最悪の場合
（極端に偏った分け方の場合）

アルゴリズム的考え方の基礎2

・ソーティング
　- クィックソート

全体の計算時間T(n)=partitionの計算時間+T(n_L)+T(n_R)
T(n_L) = partitionの計算時間+T(n_{LL})+T(n_{LR})
T(n_R) = partitionの計算時間+T(n_{RL})+T(n_{RR})
　全体の計算量➔各partitionでの基準値の選び方による



partitionの
計算時間
（最良(・平均)）
　O($n$log$_2n$)

最良の場合
（非常にバランスのとれた分け方の場合）

アルゴリズム的考え方の基礎2

・ソーティング

- ヒープソート

  計算時間 $O(n\log_2 n)$

- マージソート

  計算時間 $O(n\log_2 n)$

⇨ アルゴリズムでは、分割統治法という位置づけ
（計算量についても、分割統治法の中で詳しく解析されている）

Ex2-1 (文字列と十進数の復習)

1. char string[128]; と宣言した配列にscanf(“%s”, string);を用いてキーボードから非負の十進数を読み込み
2. int n;と宣言した整数型変数にstringが十進数として表現する整数を代入し、
3. その値を printf(“¥n%d¥n”, n); で出力し終了するプログラムを作成せよ

注意事項など

- ステップ２はatoi()やstrtod()やsscanf()を用いると一瞬で作成できるが、問題の趣旨はそういうことではないので、この課題ではprintfとscanfとfflush以外に開発環境が提供する関数を使用しないこと
- 必要な知識はプログラミング基礎と高校の授業ですべて提供されている。わからなければプログラミング基礎を復習したり、string[0]を整数としてprintfしてみよ
- 入力の値は０以上２０億以下と仮定してよい
- **入力値２０億に対し正しく動作することを必ず確認して下さい**
- 提出ファイル： ex2_1.c

Ex2-2 (ソートの計算量)

ランダムにデータ列を生成し、

- 単純なソーティング方法（選択法，交換法）における
  データ列数値の比較回数，数値の入替(Swap)回数、
- クイックソートにおける
  基準値とデータ列数値の比較回数，数値の入替(Swap)回数

を計測し、比較する.
（プログラミング基礎の講義資料とEx 39～41参照).
その際、複数回、異なるデータ列を生成して、平均を算出する.

- データ列がランダムになっているか？
- 複数回データ列を生成したときに、異なるデータ列が生成されているか？

を確認するために、以下の手順で実施すること。

Ex2-2

1）プログラミング基礎の授業資料を参考に
ソーティング方法の入力となるランダムに生成された数値からなるデータ列を作成せよ.

データ型：float型　　データの範囲：0～10000.0

要素数　：20000 (指定により変更できるようにすること)

データ列全体から**最初の10個および最後の10個**のデータを抽出・ 出力して，ランダムなデータ列の生成を確認せよ.

要素数の指定の仕方

```
#define N 20000
float data[N];
   :
   for (i=0; i<N; i++) {
         :
```

ここを変えるだけで、要素数を変更できる

★提出物　　特になし.

Ex2-2

2)　1)を指定回数繰り返し実行できるようにして，
毎回異なるデータ列が生成されるように変更せよ.
回数の指定の仕方　1) の要素数の指定の仕方を参考にせよ
提出するプログラムでは，繰り返し回数は**2回**とする

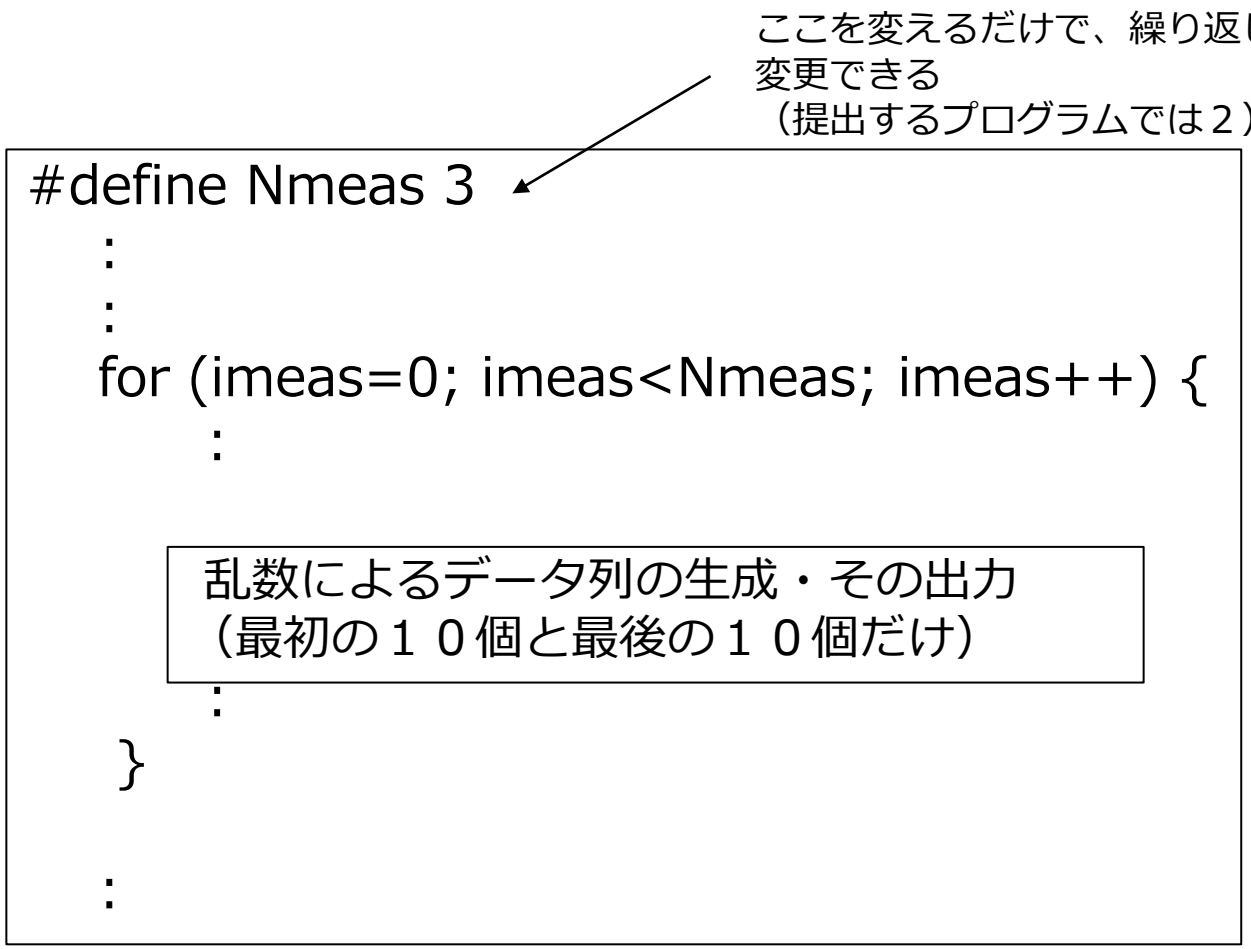
★提出物　2)のプログラム

要素数20000のランダムなデータ列の生成を2回繰り返す
1回毎に最初の10個と最後の10個のデータを出力する
（ソート不要）

“ex2_2_2.c”

＊ヒント：乱数の種(seed)の与え方を考える
生成されるデータ列の傾向が大幅に異なるように
工夫するとよい

Ex2-2（ヒント）


ここを変えるだけで、繰り返し回数を
変更できる
（提出するプログラムでは２）
#define Nmeas 3
:
:
for (imeas=0; imeas<Nmeas; imeas++) {
:
乱数によるデータ列の生成・その出力
（最初の１０個と最後の１０個だけ）
:
}
:

Ex2-2（ヒント）

プログラミング基礎の（以前の）資料より

```
float my_random(float lower, float upper){
        float r;
        r = (upper -lower)*rand()/(RAND_MAX+1.0)+lower;
// 修正
        return fabs(r);
}
// 演算誤差により[lower,upper]の範囲からはみ出すことあり
void set_data(float a[], int n, int seed){
        int i;
        srand(seed);
        for(i=0; i<n; i++){
                a[i]=my_random(0.0, NUM_RANGE);
        }
}
:
set_data(data, n, 0);
:
```

青字部分が乱数（列）の種(seed)

毎回、同じ数を使うと、毎回同じ乱数列となる

→ では、毎回違う数にするには？

Ex2-2 (ヒント)

実行毎に異なる乱数列を得るには、

```
#include <stdlib.h>
#include <time.h>
#include <math.h>
          :
    for (imeas=0; imeas<Nmeas; imeas++) {
              :
        set_data(data, n, (unsigned) time(NULL) );
              :
    }
```

但し、乱数列を生成する時刻が近いと,
殆ど同じ傾向の乱数列になります。　　→ 傾向を変更するには？

Ex2-2 (ヒント)

実行毎に異なる乱数列を得るには、

```
#include <stdlib.h>
#include <time.h>
#include <math.h>
          :
    for (imeas=0; imeas<Nmeas; imeas++) {
              :
        set_data(data, n, (unsigned) time(NULL)+imeas*1000 );
              :
    }
```

前頁の問題点に対する一つの対処法

Ex2-2

3) **単純なソーティング方法：交換法（バブルソート）** において，

・データ列の数値の比較回数，

・数値の入替(Swap)回数

を計測するプログラムを作成せよ.

- 要素数20000のデータ列をランダムに生成し，これをソートし，ソート結果の一部（→後述）およびソートにおける上記の回数を出力するプログラムを作成せよ.
- プログラムを何度か実行して，毎回データ列が異なることと，計測結果があまり変わらないことを確認すること.
- 次に，上記プログラムを拡張し，データ列の生成・ソートと回数計測・ソート結果の一部の出力を５回繰り返し，最後にソーティング１回あたりの上記の平均回数を出力するプログラムを作成せよ.
- データ列の要素数は 1)と同様に指定できるようにし、異なる要素数（例えば 60000） についても容易に計測できるようにすること.

★提出物 3)のプログラム(データ列の要素数20000）“ex2_2_3.c”

*注意：データ列の要素数を多くなると、かなり時間がかかるようになるので注意すること

更にデータ列の要素数が多くなると場合，回数を求めるときに注意が必要になる

Ex2-2

4) **高速なソーティング方法：クィックソート**（プログラミング基礎 Ex.41）において，

- 基準値とデータ列数値の比較回数，
- 数値の入替(Swap)回数

を計測するプログラムを作成せよ.

①3)と同様に，指定した要素数Nのデータ列をランダムに生成し，これをソートし，ソート結果の一部（→後述）およびソートにおける上記の回数を出力するプログラムを作成せよ.

②次に，①のプログラムを拡張して，データ列の生成・ソートと回数計測・ソート結果の一部の出力を５回繰り返し，最後にソーティング１回あたりの上記の回数を出力するプログラムを作成せよ.

★提出物　4)の拡張したプログラム（データ列の要素数20000）

“ex2_2_4.c”

3) と 4) の①の出力について

- ソート結果は，**最初の１０個**の要素および**最後の１０個**の要素のみを出力せよ
- 回数の計測結果は、数値だけではどちらかわからないので、わかるようにすること。

①の出力例

first：0.610370 1.525926 1.525926 3.051851 3.051851 3.051851 3.967406 4.272591 5.188147 6.103702

last：9996.032227 9996.337891 9996.337891 9997.252930 9997.558594 9998.167969 9998.779297 9999.083984 9999.694336 9999.694336

The number of comparisons： 123456

The number of swaps： 987654

Ex2-2

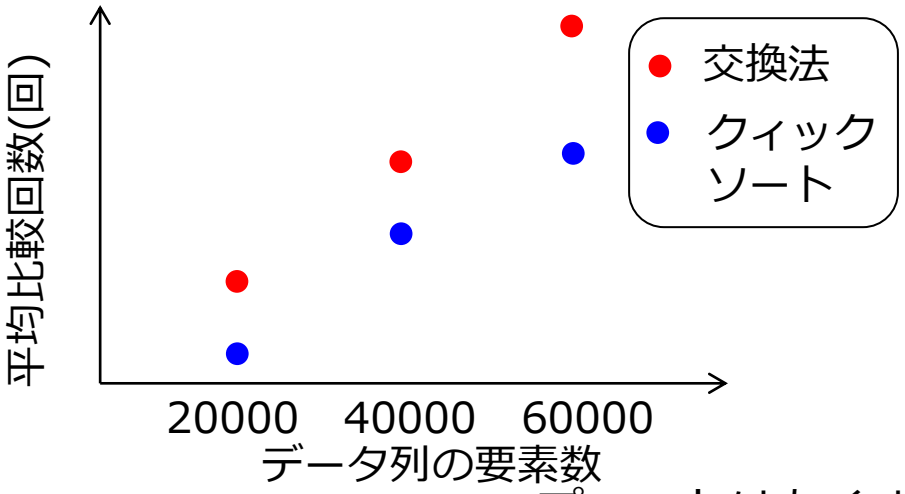
5) 3) ,4)で作成したプログラムをN=20000,40000,60000でそれぞれ実行し，得られた計測結果を表にまとめよ．また，横軸をデータ列の要素数，縦軸をそれぞれの数値（回数）として，グラフにプロットして整理してみよ．★提出物：結果をまとめた表及びプロットしたグラフ
（ファイル名: ex2_2_5.pdf）

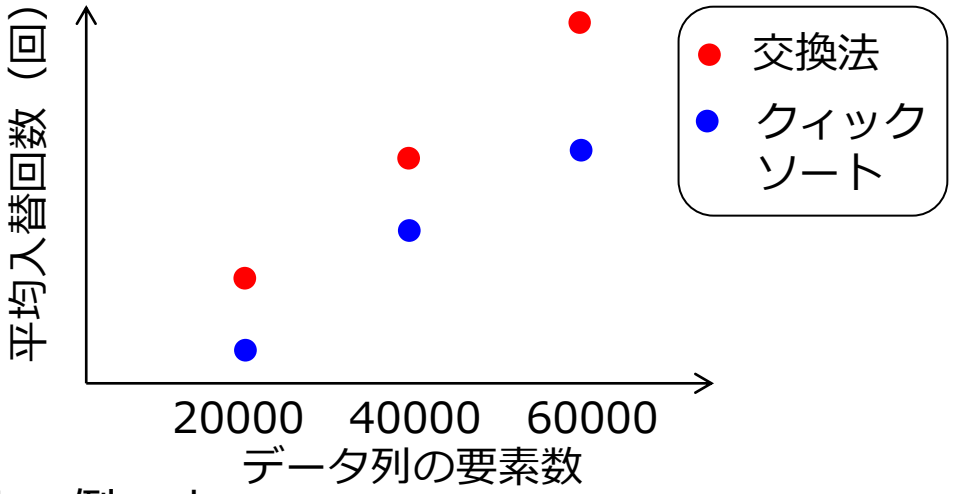
平均比較回数(回)

データ列	20000	40000	60000
交換法			
クィックソート			

平均入替回数(回)

データ列	20000	40000	60000
交換法			
クィックソート			





プロットはあくまで例です。

Ex2-2

6) （発展的課題:余裕のある人向け）

3)と同様に “単純なソーティング方法：選択法” について,

・データ列の数値の比較回数,

・数値の入替(Swap)回数

を計測するプログラムを作成せよ.

(対象データ列の要素数は20000, 40000, 60000)

また，マージソート（プログラミング基礎Ex.42）について,

4) と同様に

・ データ列数値の比較回数,

を計測するプログラムを作成せよ.

(対象データ列の要素数は20000, 40000, 60000．計測は対象データ列を変えて５回繰り返し，ソーティング１回あたりの数値を出力する)

★提出物　特になし

計測結果を，5) の表及びプロットに反映した場合は加点対象とします

今日の課題の提出物の一覧

学籍番号や氏名はファイル名に含めないで下さい

- ex2_1.c
- ex2_2_2.c
- ex2_2_3.c
- ex2_2_4.c
- ex2_2_5.pdf
- 前回に課題１－１解答を提出していない場合、上記ファイルと一緒に第2回課題のコーナーに提出をお願いします

注意

Windowsではマージソートで要素数Nを増やすと「関数内の変数で使えるメモリ量の上限（次ページで補足）」に達してうまく動かなくなる可能性がある。

回避方法としては、

- プログラムを工夫して消費メモリ量を減らす
- ある再帰関数 F 内の配列を、F の複数の再帰呼び出しで共用する（詳細は次回説明）

などがあるが、現時点では、

- **計算機室のMacOSを使う**

が手っ取り早い

関数の中でstaticを付けないで宣言した配列について

- そのような配列で確保できる要素数の（再帰呼び出しを含むすべての関数呼び出しにおける）合計には上限があります。
- ウィンドウズではだいたいfloat型の整数25万個程度だったはずです（それ以外だともっと上限が大きいことが多い）
- static については次回できちんと説明します

お願い

採点に使うMacOSまたはLinuxではrandom()という関数が定義されている。提出するプログラムで独自のrandom()を定義するとエラーが起きるので定義しないでください（間違いではないから減点はしない）

うまく動かない質問時のお願い

- プログラムが上手く動作していないと疑わしい部分の変数を printf で表示するようにしてから質問して下さい（課題提出時にはデバッグ用printfは消す）
- （講師が松本のときのみ）質問する前に「check1.sh」によるコンパイラの警告と実行時エラーを無くして下さい。警告やエラーの意味がわからないときは気軽に聞いてください

重要：コピペレポートについて

プログラムや考察などが他の提出者と重複している場合、不正とみなして減点および問い合わせをすることがあります

未定義動作

- 配列の範囲外アクセスや、未初期化変数からの読み出しを行うと、C言語規格では**動作が未定義**で「何が起きるか全く保証できない（異常終了するかもしれない）」と書いてあります。
- 自動採点システムではこれら未定義動作を検出し、未定義動作はすべて減点します。

未定義動作を調べる検査

ターミナルから ~matsumoto.r.aa/bin/check1.sh ナントカ.c を実行して未定義動作を見つけることができる