情報処理論2a

3章「整列と計算量」

(バブルソートとクイックソート)

中島康彦

§3. 1 今日の作業ディレクトリを作る

1. % cd ⇒ ホームディレクトリへ移動
2. % mkdir chap15 ⇒ ディレクトリchap15を作成
3. % cd chap15 ⇒ ディレクトリchap15へ移動
4. % netscapeを使ってdata15をchap15へダウンロード
5. % tar xvf data15 ⇒ サンプルデータの複写

```
sort.bubble.c
sort.quick.c
sort.data
sort.data.70000
```

§3. 2 整列(ソート)

ある企業が各都道府県に保有している店舗の売場面積

```
234.5228 北海道
 92.3446 青森県
132.7822 岩手県
 68.6096 宮城県
114.3411 秋田県
 73.9433 山形県
130.8248 福島県
 60.9378 茨城県
 64.0828 栃木県
 63.6316 群馬県
 37.6709 埼玉県
 49.9572 千葉県
 21.0214 東京都
```

§3. 3 データの作成(sort.data)

浮動小数点数の計算は，整数より低速.
必要のない限り，浮動小数点数は使わない.
一律10000倍することにより，整数演算にできる.

```
2345228 北海道
 923446 青森県
1327822 岩手県
 686096 宮城県
1143411 秋田県
 739433 山形県
1308248 福島県
 609378 茨城県
 640828 栃木県
 636316 群馬県
 376709 埼玉県
 499572 千葉県
 210214 東京都
```

§3. 4 機能設計と構成設計

機能設計(入出力形式)

扱うことのできる都道府県数の上限　…200
扱うことのできる都道府県名の最大長　…79
標準入力　…売場面積，都道府県名
標準出力　…整列結果(狭い順)

構成設計(内部データ構造)

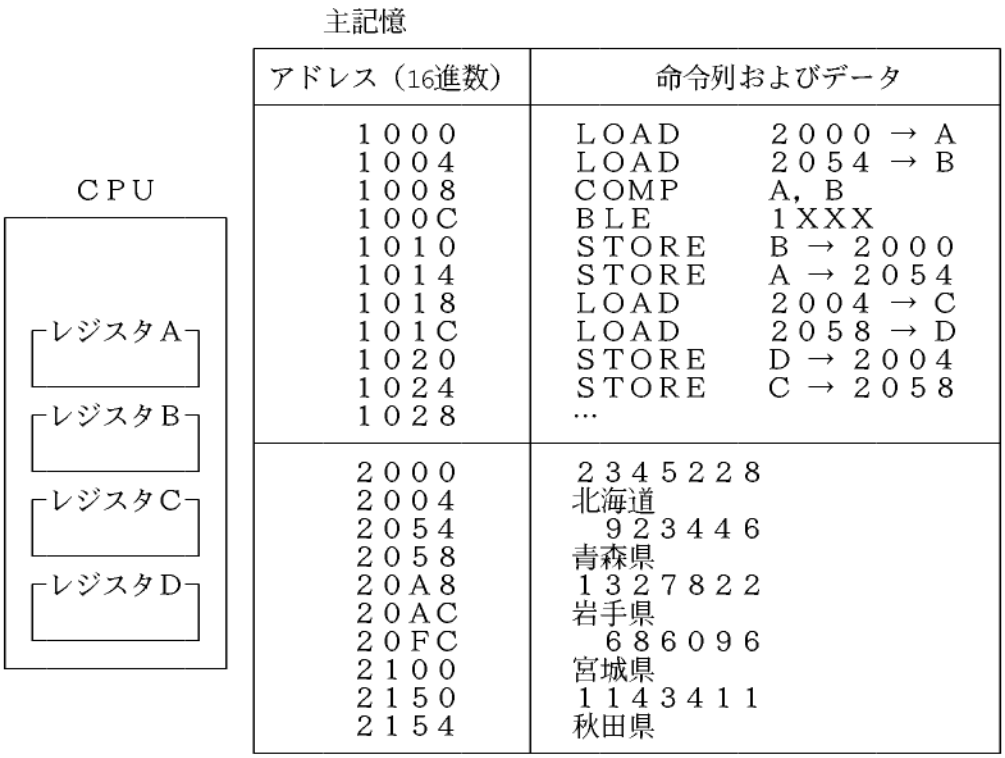
都道府県データを記憶する配列　…pd[200]
1要素に売場面積と名前を記憶　…struct pref pd[200]
　売場面積を記憶する変数　…int型 area
　都道府県名を記憶する変数　…char型 name[80]

```
struct pref {int area; char name[80];} pd[200]
```

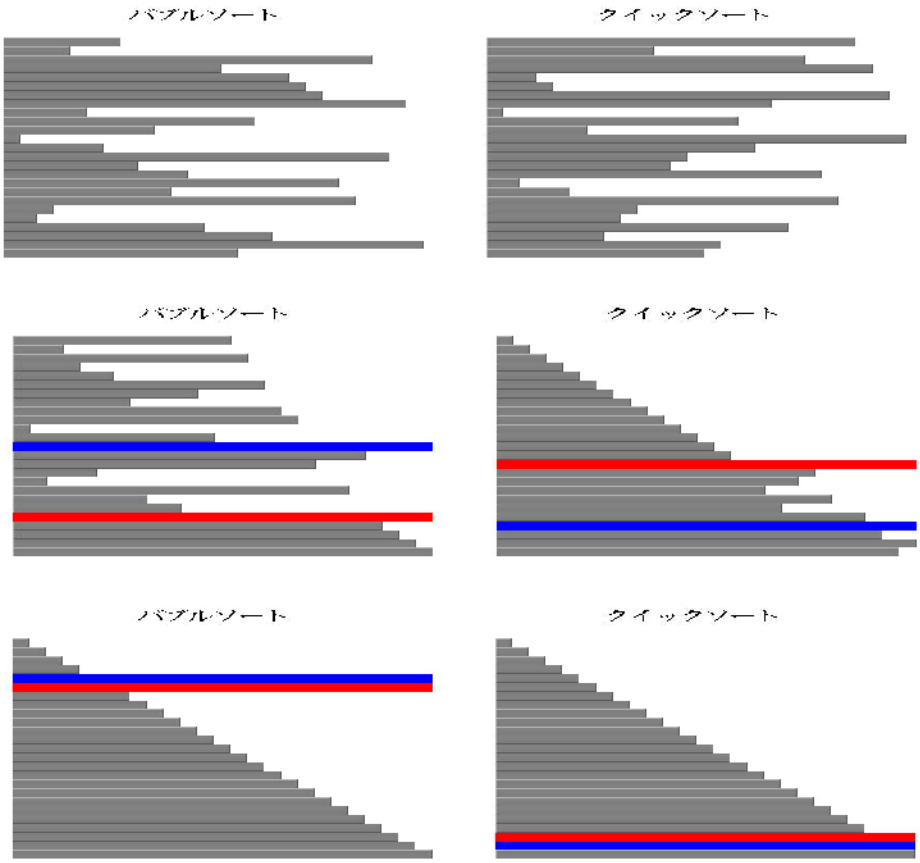
都道府県数を記憶する変数　…int型 size

§3. 5 コンピュータの動作と計算量

実行する命令数が少ないほど高速



§3. 6 アルゴリズムの違いと計算量



§3. 7 バブルソート

隣合う2つを順に比較し，大きい方を下に移動する.

(1) 上から順に4回の比較を行う.

北海道	2345228★←	青森県	923446	青森県	923446
青森県	923446★←	北海道	2345228★←	岩手県	1327822
岩手県	1327822 交換	岩手県	1327822★←	宮城県	686096
宮城県	686096	宮城県	686096 交換	秋田県	1143411
秋田県	1143411	秋田県	1143411	北海道	2345228■

(2) 再度，上から順に3回の比較を行う.

青森県	923446★—	青森県	923446	青森県	923446
岩手県	1327822★—	岩手県	1327822★←	宮城県	686096
宮城県	686096	宮城県	686096★←	秋田県	1143411
秋田県	1143411	秋田県	1143411 交換	岩手県	1327822■
北海道	2345228	北海道	2345228	北海道	2345228

§3. 7 バブルソート(続き)

(3) 再度, 上から順に2回の比較を行う.

青森県	923446★←	宮城県	686096	宮城県	686096
宮城県	686096★←	青森県	923446★—	青森県	923446
秋田県	1143411	交換	秋田県	1143411★—	秋田県	1143411■
岩手県	1327822		岩手県	1327822	岩手県	1327822
北海道	2345228		北海道	2345228	北海道	2345228

(4) 最後に, 上から順に1回の比較を行う.

宮城県	686096★—	宮城県	686096
青森県	923446★—	青森県	923446■
秋田県	1143411	秋田県	1143411
岩手県	1327822	岩手県	1327822
北海道	2345228	北海道	2345228

合計 10回の比較と7回の交換
47都道府県の場合 1081回/654回

§3. 8 詳細設計(sort.bubble.c)

```
#include <stdio.h>
#define MAXSIZE 200
#define MAXNAME 80

struct pref {
  int  area;
  char name[MAXNAME];
} pd[MAXSIZE];

struct pref *pP[MAXSIZE];

int size;
int compare;
int swap;

main()
{
  int i;

  readfile();
  bsort();                                          /* 整列の呼び出し */
  for (i = 0; i<size; i++)
    printf("%8d %s\n", pP[i]->area, pP[i]->name);   /* 整列結果を表示 */
  printf("compare=%d swap=%d\n", compare, swap);    /* 比較/交換回数を表示 */
}

readfile()
{
  for (size=0; size<MAXSIZE; size++) {
    pP[size] = &pd[size];
    if (scanf("%d %80s", &pd[size].area, &pd[size].name[0]) != 2)
      break;
  }
}
```

§3. 8 詳細設計(sort.bubble.c 続き)

```
bsort()                                                 /* 整列 */
{
  int i, j;
  struct pref *tp;

  for (i = size-1; i>=0; i--) {
    for (j = 0; j<i; j++) {
      compare++;
      if (pP[j]->area > pP[j+1]->area) {                /* 売場面積の比較 */
        tp = pP[j]; pP[j] = pP[j+1]; pP[j+1] = tp;      /* ポインタの交換 */
        swap++;
      }
    }
  }
}
```

§3. 9 解説

```
#include <stdio.h>
#define MAXSIZE 200
#define MAXNAME 80
```

cプリプロセッサの処理対象となる定義

- システムファイル取り込み
- 定数の定義(コンパイル前に識別子を置換する)

  MAXSIZE ⇒ 200

  MAXNAME ⇒ 80

§3. 9 解説(続き)

```
struct pref {int area;char name[MAXNAME];} pd[MAXSIZE];
```

構造体の定義

- 構造体タグ(構造体の構成)prefの定義
  ```
  struct pref {...};
  ```
- struct prefの1単位は1つの整数と1つの文字配列
  ```
  {int  area; char name[MAXNAME];}
  ```
- 構造体の実体であるpdの定義
  ```
  struct pref {...} pd;
  ```
- 構造体を要素とする配列の場合は
  ```
  struct pref {...} pd[MAXSIZE];
  ```

外部変数

- 関数の外側で定義すると外部変数(関数間で共有)
- 初期値指定が無いので，初期値は0

§3. 9 解説(続き)

```
struct pref *pP[MAXSIZE];
```

ポインタの定義

- 構造体タグprefは既定義
- prefの構造を有する構造体へのポインタpPの定義
  ```
  struct pref *pP;
  ```
- ポインタを要素とする配列の場合は
  ```
  struct pref *pP[MAXSIZE];
  ```

外部変数

- pP[MAXSIZE]も外部変数
- 初期値指定が無いので，全ての要素の初期値はNULLポインタ(数値としては0)
- NULLポインタのまま使用するとプログラムが異常終了

§3. 9 解説(続き)

```
int size;
int compare;
int swap;
```

外部変数の宣言

- 整数型(32ビット)
- 初期値指定が無いので，初期値は0

`size:` データ数計測用
`compare:` 比較回数計測用
`swap:` 交換回数計測用

§3. 9 解説(続き)

```
main()
{
  int i;

  readfile();
  bsort();
  for (i = 0; i<size; i++)
    printf("%8d %s\n", pP[i]->area, pP[i]->name);
  printf("compare=%d swap=%d\n", compare, swap);
}
```

内部変数の宣言

- 初期値指定が無いので，初期値は不定

関数`readfile()`によるファイルからのデータ読み込み
関数`bsort()`による整列
関数`printf()`による整列結果の出力
同様に，比較回数/交換回数の出力

§3. 9 解説(続き)

```
readfile()
{
  for (size=0; size<MAXSIZE; size++) {
    if (scanf("%d %80s", &pd[size].area,
                         &pd[size].name[0]) != 2)
      break;
    pP[size] = &pd[size];
  }
}
```

for文により，最大でMAXSIZE分のデータを読み込む

- 売場面積をint型変数pd[size].areaへ
- 都道府県名をchar型配列pd[size].name[]へ
- 代入失敗時は，データ読み込み終了
- pP[]の各要素にpd[]各要素のアドレスを代入

```
pP[size] = &pd[size];
```

§3. 9 解説(続き)

```
bsort()
{
  int i, j; struct pref *tp;

  for (i = size-1; i>=0; i--)
    for (j = 0; j<i; j++) {
      compare++;
      if (pP[j]->area > pP[j+1]->area) {
        tp = pP[j]; pP[j] = pP[j+1]; pP[j+1] = tp;
        swap++;
      }
    }
}
```

バブルソートによる整列

- 構造体の実体pdを交換するのではなく，ポインタppを交換する(移動量が小さい)
- 交換のために第三の変数tpを用いる.
- 構造体メンバの参照は

  実体の場合: pd[j].area

  ポインタの場合:pP[j]->area
- 比較回数/交換回数を変数compare/swapに記録

§3. 10 クイックソート

中央付近の値を基準値とする.

上から基準値以上のもの，下から基準値以下のものを探し，最初の要素どうしを交換する.

これを繰り返すと，上半分に基準値以下のもの，下半分に基準値以上のものが集まる.

上半分，下半分の各々について同様に繰り返す.

§3. 10 クイックソート(続き)

(1) 中央位置の値(1327822)を基準値とする.

北海道	2345228★←	秋田県	1143411	秋田県	1143411
青森県	923446	青森県	923446	青森県	923446
岩手県	1327822—基準	岩手県	1327822★←	宮城県	686096
宮城県	686096	宮城県	686096★←	岩手県	1327822
秋田県	1143411★←	北海道	2345228 交換	北海道	2345228
	交換				

(2) 最後の注目点を境に，上半分，下半分について同様に繰り返す.

秋田県	1143411★←	宮城県	686096
青森県	923446 —	青森県	923446★←
宮城県	686096★←	秋田県	1143411 (交換)
	交換		
岩手県	1327822★—	岩手県	1327822★←
北海道	2345228★—	北海道	2345228 (交換)

合計 11回の比較と5回の交換

47都道府県の場合 298回/77回

§3. 11 詳細設計(sort.quick.c)

```
main()
{
  :途中同じ
  qsort(0, size-1);                                    /* 整列の呼び出し */
  :途中同じ
}

readfile()
{
  :途中同じ
}

qsort(int lo, int hi)                                  /* 整列 */
{
  int i = lo, j = hi, ref;
  struct pref *tp;

  ref = pP[(i + j)/2]->area;                           /* 中間位置の売場面積を基準値とする */
  while (i <= j) {
    while ((i < hi) && (compare++,pP[i]->area < ref))/* 基準値以上の最初の県 */
      i++;
    while ((j > lo) && (compare++,pP[j]->area > ref))/* 基準値以下の最初の県 */
      j--;
    if (i <= j) {                                      /* 交換するか */
      tp = pP[i]; pP[i] = pP[j]; pP[j] = tp;           /* ポインタの交換 */
      i++;
      j--;
      swap++;
    }
  }
  if (lo < j) qsort(lo, j);                            /* 上半分を整列 */
  if (i < hi) qsort(i, hi);                            /* 下半分を整列 */
}
```

§3. 12 解説

```
qsort(int lo, int hi)
{
  int i = lo, j = hi, ref;
  struct pref *tp;

  ref = pP[(i + j)/2]->area;
  :
}
```

クイックソートによる整列

- 引数loからhiの範囲の配列要素を整列する.
- 最初に変数iにlo，jにhiを代入する.
- 中央付近の値を基準値 ref とする.

§3. 12 解説(続き)

```
while (i <= j) {
  while ((i < hi) && (compare++,pP[i]->area < ref)) i++;
  while ((j > lo) && (compare++,pP[j]->area > ref)) j--;
  if (i <= j) {
    tp = pP[i]; pP[i] = pP[j]; pP[j] = tp;
    i++; j--; swap++;
  }
}
if (lo < j) qsort(lo, j);
if (i < hi) qsort(i, hi);
```

- loから下へpP[i]->area ≧ ref,
- hiから上へpP[j]->area ≦ refを探し,
- i≦jであれば要素pP[i]とpP[j]を交換
- lo≦j<i≦hiとなるまで繰り返す.
- 上半分lo～j，下半分i～hiの各々を整列
  自身を呼び出す(再帰的呼出し)

§3. 13 コンパイルと実行

1．コンパイルする.

```
% gcc sort.bubble.c -o sort.bubble
% gcc sort.quick.c  -o sort.quick
```

2．実行

```
% ./sort.bubble < sort.data または
% ./sort.quick  < sort.data
```

```
186146 香川県
189276 大阪府
210214 東京都
226788 沖縄県
241511 神奈川県
243918 佐賀県
280143 富山県
350708 鳥取県
369109 奈良県
376709 埼玉県
:
compare=1081 swap=654 (sort.bubbleの場合)
```

§3. 14 例題

都道府県数は47であるが，以上のプログラムは200組までのデータを整列可能である．catコマンドを併用して1~4倍の大きさのデータに対してプログラムを走行し，以下の問いに答えよ．

```
% cat sort.data                                    | ./sort.xxx
% cat sort.data sort.data                          | ./sort.xxx
% cat sort.data sort.data sort.data                | ./sort.xxx
% cat sort.data sort.data sort.data sort.data      | ./sort.xxx
```

- compareとswap各々について，クイックソートに対するバブルソートの値が何倍に変化したかを計算せよ．

- データ数が16倍である場合に，クイックソートに対するバブルソートの値が何倍であると推測されるか．また根拠を述べよ．

§3. 15 今日の課題

sort.data.70000は，70000行(1MB)のファイルである．Excelによりソートすることはできない．70000行のファイルを扱えるようsort.bubble.cとsort.quick.cを改造し，compare回数とswap回数の比を測定せよ．

- 70000に変えただけではcompare回数がおかしい
- 47行の時，バブルソートのcompare回数は(47*46)/2

宛先: nakashim@econ.kyoto-u.ac.jp
件名: unix2-学生番号

今日はここまで