

アルゴリズム論 8

整列処理(ソート)

- バブルソート
- 単純選択ソート
- 挿入法
- クイックソート
- ヒープソート

単純選択ソート(最小値選択法)

以下のテストの点数を昇順に並べなさい

	1	2	3	4	5	
手順1	60	75	70	56	52	最小値を見つける
手順2	52	75	70	56	60	1番目の数値と最小値を交換
手順3	52	75	70	56	60	2番目以降の最小値を見つける
手順4	52	56	70	75	60	2番目の数値と最小値を交換
手順5	52	56	70	75	60	3番目以降の最小値を見つける
手順6	52	56	60	75	70	3番目の数値と最小値を交換
手順7	52	56	60	75	70	4番目以降の最小値を見つける
手順8	52	56	60	70	75	4番目の数値と最小値を交換

完了！

単純選択ソート1(メイン)

```
#include <stdio.h>

#define swap(type,x,y) {type t=x; x=y; y=t;}

#define NUM 5
    /* count0:比較回数, count1:交換回数 */
int count0=0,count1=0; /* グローバル変数として初期化 */

void selsort(int a[], int n); /* 関数プロトタイプ */
```

単純選択ソート2(メイン)

```
int main(void)
{
    int    i;
    int          x[NUM];

    printf("Input integer number %d times ¥n", NUM);
    for (i=0; i<NUM; i++)    {
        printf("x[%d]:", i);
        scanf("%d", &x[i]);
    }
    selsort(x, NUM);
    printf("Sorting is finished ¥n");
    for (i=0; i<NUM; i++)
        printf("x[%d] =%d¥n", i, x[i]);

    printf("Number of comparison=%d¥n", count0);
    printf("Number of swap=%d¥n", count1);
    return(0);
}
```

単純選択ソート3(関数)

```
void selsort(int a[], int n)
{
    int i,j,min;

    for (i=0;i<=n-1;i++) {
        min=i;
        for (j=i+1;j<=n-1;j++) {
            if (a[j]<a[min]) {
                count0++;
                min=j;
            } else count0++;
        }
        swap (int, a[i],a[min]);
        count1++;
    }
}
```

単純選択ソート 実行結果

Input integer number 5 times

x[0]:60

x[1]:75

x[2]:70

x[3]:56

x[4]:52

Sorting is finished

x[0] =52

x[1] =56

x[2] =60

x[3] =70

x[4] =75

Number of comparison=10

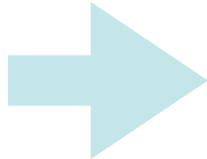
Number of swap=5

単純選択ソートの計算量

n個のデータのソート

- 比較回数

– $(n-1)+(n-2)+\cdots+2+1$ 回

– $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ 回  オーダ $O(n^2)$

- 交換回数

– n回

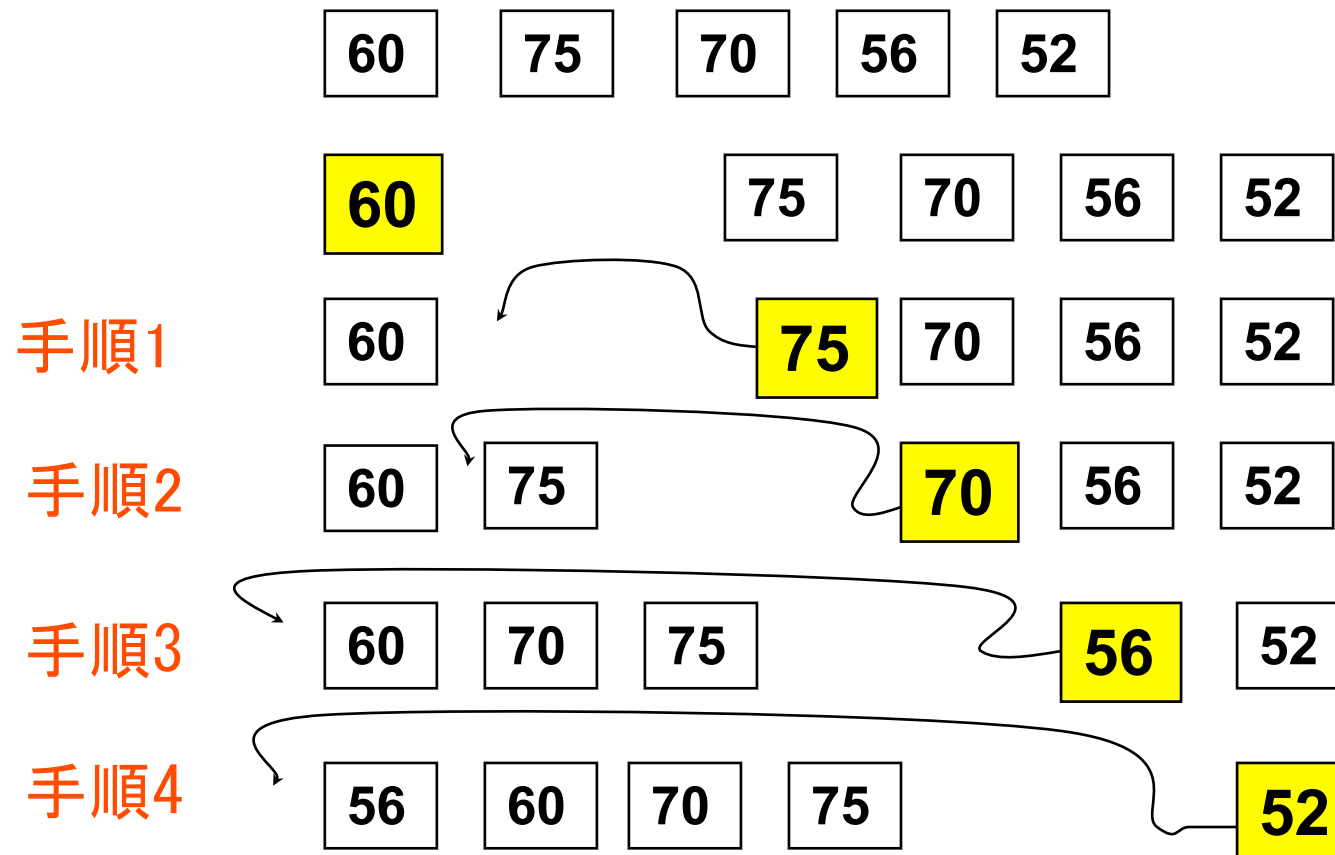
演習問題(講義時間内で実施)

- ソートを行うプログラムのソースコードを入力し実行する
 - メイン
 - 単純選択法 関数
- データを入力し、実行結果を確認する

挿入法とは

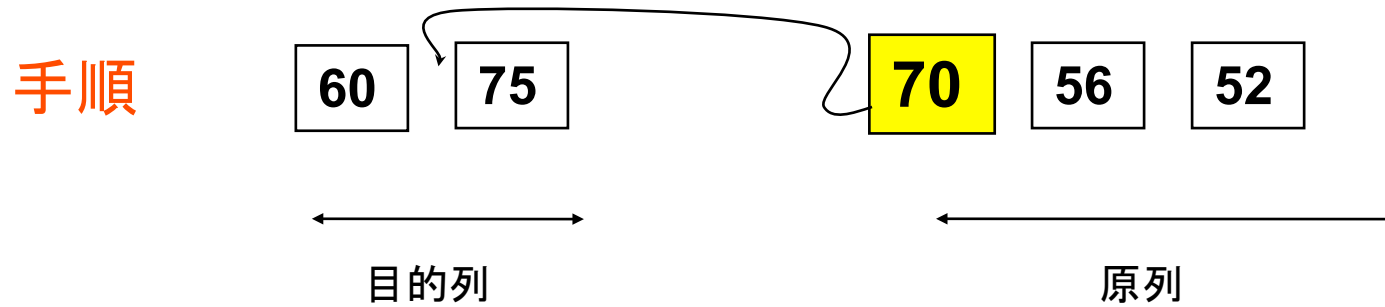
- 挿入法: バブルソートよりは高度な処理であるが、きわめて普通の方法である

以下のテストの点数を昇順に並べなさい



完了!

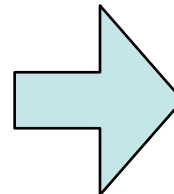
手順に着目してアルゴリズムを考える



目的列
(ソート済) $a[0], a[1]$

原列 $a[2], a[3], a[4]$

1. $a[2]$ を $a[1]$ と比較して大きければ $a[1]$ の右に入れる。小さければ、次の $a[0]$ と比較する。
2. 目的列の左端に達したかどうかを判断する。
3. 挿入場所より右のデータについて要素を一つずつ増加させる。



1. $a[j]$ を $a[j-1]$ と比較して大きければ $a[j-1]$ の右に入れる。小さければ、次の $a[j-2]$ と比較する。
2. 目的列の左端に達したかどうかを判断する。 $(j > 0)$
3. 挿入場所より右のデータについて要素を一つずつ増加させる。

$(a[j] = a[j-1] \dots)$

挿入法プログラム1(メイン)

```
#include <stdio.h>

#define NUM 5
    /* count0:比較回数, count1:挿入回数 */
int count0=0,count1=0; /* グローバル変数として初期化 */

void inssort(int a[], int n); /* 関数プロトタイプ */
```

挿入法プログラム2(メイン)

```
int main(void)
{
    int        i;
    int        x[NUM];

    printf("Input integer number %d times ¥n", NUM);
    for (i=0; i<NUM; i++) {
        printf("x[%d]:", i);
        scanf("%d", &x[i]);
    }
    inssort(x, NUM);
    printf("Sorting is finished ¥n");
    for (i=0; i<NUM; i++)
        printf("x[%d] =%d¥n", i, x[i]);

    printf("Number of comparison=%d¥n", count0);
    printf("Number of insertion=%d¥n", count1);

    return(0);
}
```

挿入法プログラム3(関数)

```
void inssort(int a[], int n)
{
    int i, j, tmp;
    for (i=1; i<=n-1; i++) {
        tmp=a[i];
        j=i;
        while ((a[j-1]>tmp) && (j>0)) {
            a[j]=a[j-1];
            j--;
            count0+=2; /* 比較 */
        }
        a[j]=tmp;
        count1++; /* 挿入 */
        count0+=2; /* 比較 */
    }
}
```

挿入法実行結果

Input integer number 5 times

x[0]:60

x[1]:75

x[2]:70

x[3]:56

x[4]:52

Sorting is finished

x[0] =52

x[1] =56

x[2] =60

x[3] =70

x[4] =75

Number of comparison=24

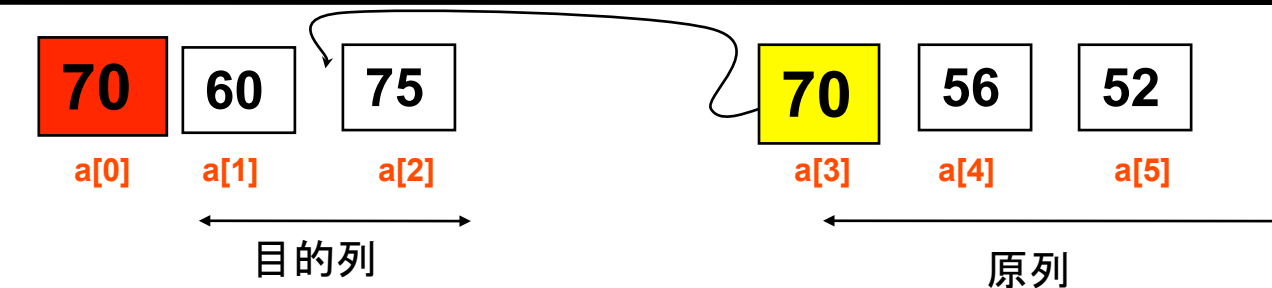
Number of insertion=4

番兵法(sentinel)とは

- 番兵法(sentinel):条件の見張り
 - 繰り返しの終了判定を簡略化するためにデータを追加する。
 - 通常データ列の一番始めか一番後ろに追加
- 挿入法の場合
 - データ列の一番始めに番兵を入れる
 - **a[0]**を利用する
- 番兵法の効果
 - **比較の回数を減らす**ことが可能

番兵法の効果

手順



目的列
(ソート済) **$a[1], a[2]$**

原列 **$a[3], a[4], a[5]$**

通常の挿入法

1. $a[j]$ を $a[j-1]$ と比較して大きければ $a[j-1]$ の右に入れる。小さければ、次の $a[j-2]$ と比較する。
2. 目的列の左端に達したかどうかを判断する。 $(j > 0)$
3. 挿入場所より右のデータについて要素を一つずつ増加させる。

$(a[j] = a[j-1] \dots)$

番兵法を使用した挿入法

$a[0]$ として挿入する値を入れることによって挿入する値($a[1] \sim a[5]$)は必ず $a[0]$ に等しい。

$a[0]$ より右のどこかへ挿入できる!!!

1. $a[j]$ を $a[j-1]$ と比較して大きければ $a[j-1]$ の右に入れる。小さければ、次の $a[j-2]$ と比較する。
2. 挿入場所より右のデータについて要素を一つずつ増加させる。

$(a[j] = a[j-1] \dots)$

番兵法を使用した挿入法関数

```
void sent_inst(int a[], int n)
{
    int    i, j;
    for (i=2; i<=n; i++) {
        a[0]=a[i];
        j=i;
        while (a[j-1]>a[0]) {
            a[j]=a[j-1];
            j--;
            count0++;    /* 比較 */
            count1++;    /* 交換 */
        }
        a[j]=a[0];
        count2++;        /* 挿入 */
        count0++;        /* 比較 */
    }
}
```

番兵法を使用した挿入法実行結果

Input integer number 5 times

x[1]:60

x[2]:75

x[3]:70

x[4]:56

x[5]:52

Sorting is finished

x[1] =52

x[2] =56

x[3] =60

x[4] =70

x[5] =75

Number of comparison=12

Number of insertion=4

番兵法を使用しない場合より

比較の回数が $1/2$ になる

演習問題(講義時間内で実施)

- ソートを行うプログラムのソースコードを入力し実行する
 - メイン
 - 挿入法 関数
- データを入力し、実行結果を確認する

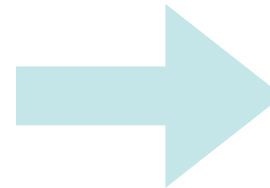
挿入法 の計算量

n個のデータのソート

- 比較回数
 - whileループの反復回数

- 最小：n-1回

- 最大： $\sum_{i=2}^n i = \frac{n(n+1)}{2} - 1$ 回



オーダー $O(n^2)$

- 挿入回数
 - n-1回