

アルゴリズム論 7

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

整列処理(ソート)とは

データの整列処理（並び替え）：Sorting

データを一定の基準に従って並び替えること

大量のランダムに並んだデータ

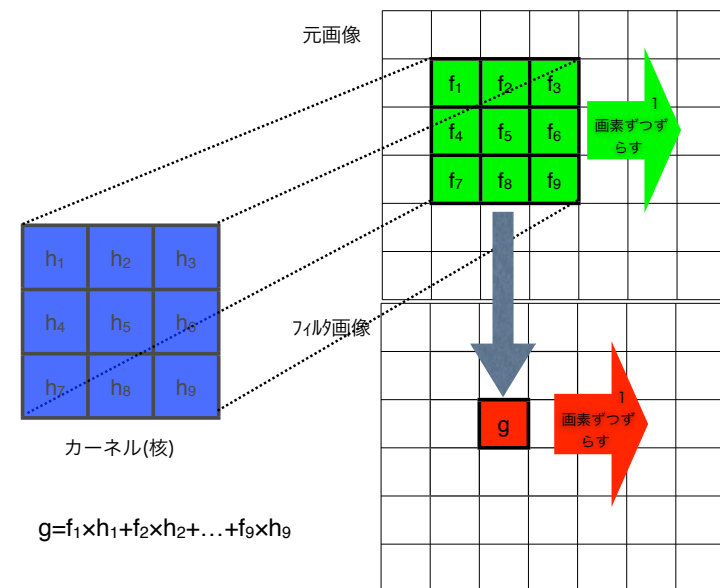
基準：降順(大きい順) or 昇順(小さい順)
 descending order ascending order

昇順または降順に並んだデータ

整列処理の応用分野1

- データベースの並び替え
 - 住所録の氏名の並び替え
 - あいうえお順
 - アルファベット順
 - 生年月日順
 - 学生の成績データベース
 - 成績順
- 信号処理、画像処理への適用
 - メジアンフィルタ：
 - ノイズを除去する目的で画像窓中の中間値を出力する

デジタル画像のフィルタリング処理



デジタル画像のフィルタリング処理(平滑化)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

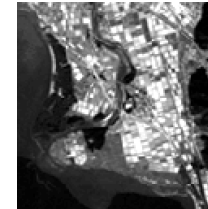
平均値

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

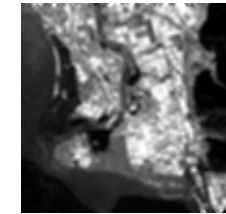
メディアン

- $f_1 \sim f_9$ のメディアン(中央値)をとる
- 画像データの整列処理が必要
- 非線形のフィルタである

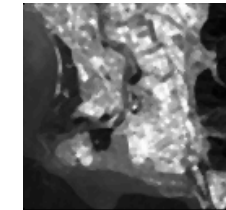
デジタル画像のフィルタリング処理(平滑化)



オリジナル画像



平均値

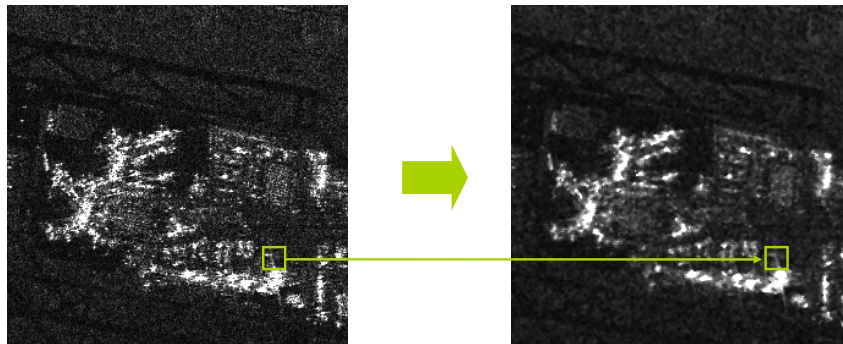


メディアン

整列処理の応用分野2

アルゴリズム論 ソート

メディアンフィルタ適用例(レーダ画像)



カナダ バンクーバ空港

メディアン(中央値)

データを昇順に並べたときに、真中に来る値。データ数が偶数のときは2つの平均値となる。

整列処理の種類

アルゴリズム論 ソート

- 整列処理の基本
 - 交換
 - 選択(比較)
 - 挿入
- 整列処理の種類
 - バブルソート(単純交換ソート): bubble sort
 - ✓ 隣同士の比較、交換だけでソートする
 - 単純選択ソート: selection sort
 - ✓ データ列の最小値を選択し、未整列部分の先頭に置く
 - 挿入法: insertion sort
 - ✓ ソートするデータをしかるべき場所に挿入
 - ✓ 少ないデータ、ほぼソートの終わったデータに有効
 - クイックソート: quick sort
 - ✓ 分割統治法を応用
 - ✓ 平均的には最も速いソート
 - ヒープソート: heap sort
 - ✓ ヒープと呼ばれるデータ構造を応用
 - ✓ 比較的早いソート

バブルソート(単純交換ソート)

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
手順2	60	75	70	52	56
手順3	60	75	52	70	56
手順4	60	52	75	70	56
手順5	52	60	75	70	56
手順6	52	60	75	56	70
手順7	52	60	56	75	70
手順8	52	56	60	75	70
手順9	52	56	60	70	75
手順10	52	56	60	70	75

手順:

隣り合うデータを比較し、左>右の場合は交換する。

比較回数
4+3+2+1=10

完了!

バブルソートプログラム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 bubble(int a[], int n); /* 関数プロトタイプ */
```

バブルソートプログラム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]);
    }
    bubble(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);
}
```

バブルソートプログラム3(関数)

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

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

隣り合うデータを比較し、左>右の場合は交換する。

バブルソート実行結果

```

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=8

```

降順にデータをソートするための修正点は？

13


演習問題7-1(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
 - バブルソート 関数
- データを入力し、実行結果を確認する

14

バブルソートの計算量

n個のデータのソート

- 比較回数
 - $(n-1)+(n-2)+\dots+2+1$ 回
 - $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ 回  オーダ $O(n^2)$
 - 比較回数を減らす工夫が可能
 - あるパスで交換が起きなくなった場合：

以降のパスは比較を省略できる
- 交換回数
 - 初期のデータ並び順によって異なる

15