

# データ構造入門及び演習

## 8回目：整列処理(挿入法)

---

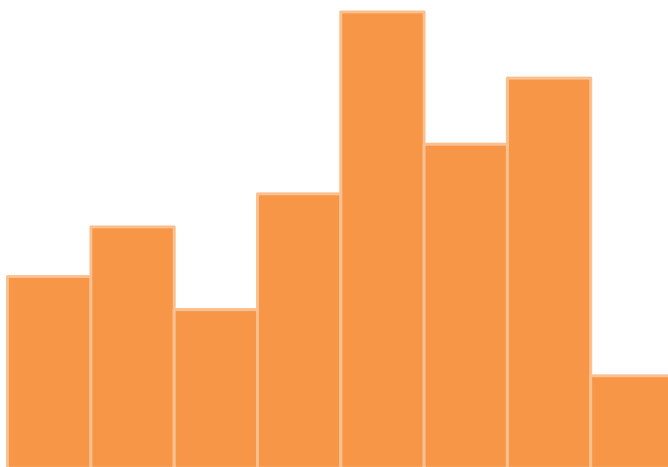
2014/06/06

担当：見越 大樹

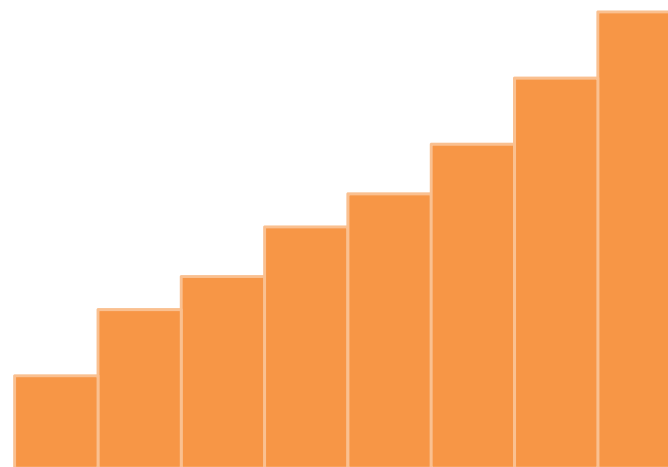
61号館304号室

# ソート (sorting)

- ソートとは？
  - データの整列処理（並び替え）
  - データを一定の基準に従って並び替えること（大きい順, 小さい順, など）



整列前データ



整列後データ

# ソートの応用分野

- 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
  - 学生のデータベースの並び替え
    - 学生番号順
    - 成績順

# 整列処理の種類

- 整列処理を行う方法には、様々な種類がある
  - バブルソート(単純交換ソート) 本講義で学習(復習)
  - 挿入法(単純挿入法) 本講義で学習
  - クイックソート 本講義で学習
  - ヒープソート
- 整列の3つの基本処理:
  1. 比較
  2. 交換
  3. 挿入

# バブルソート

- バブルソートとは？
  - 小さい順に並び替える場合,
  - 基本処理:隣り合う要素を比較して, 右のほうが小さければ両者を入れ替える
  - 基本処理を左から順に行い, 大きい数を右に寄せる

# バブルソート

手順： 1と2を繰り返す  
1. 隣り合う要素を比較  
2. 左 > 右なら交換

初期

a[0]	a[1]	a[2]	a[3]	a[4]
60	75	70	56	52



最終

52	56	60	70	75
----	----	----	----	----

# バブルソート

手順： 1と2を繰り返す  
1. 隣り合う要素を比較  
2. 左 > 右なら交換

初期

a[0]	a[1]	a[2]	a[3]	a[4]
60	75	70	56	52
↓	↓			
60	75	70	56	52

1. そのまま

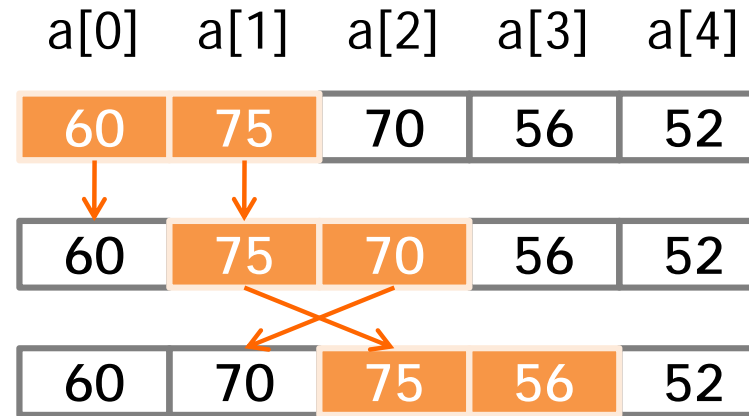
最終

52	56	60	70	75
----	----	----	----	----

# バブルソート

手順： 1と2を繰り返す  
1. 隣り合う要素を比較  
2. 左 > 右なら交換

初期



1. そのまま
2. 入れ替え

最終

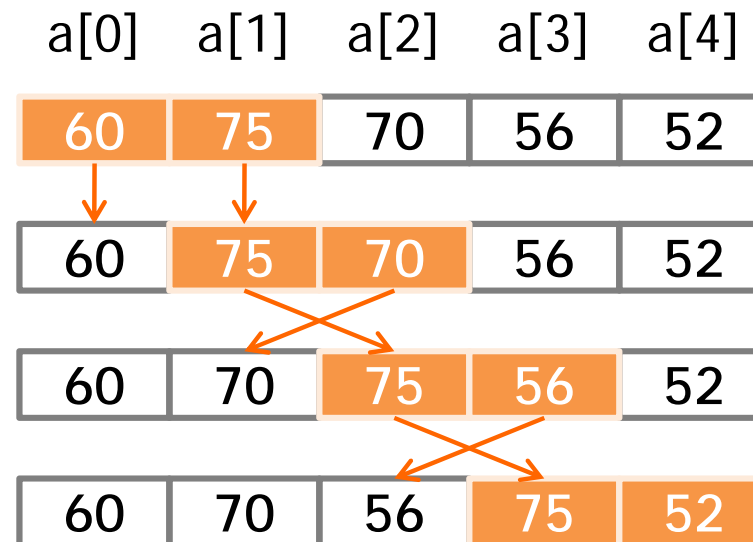
52	56	60	70	75
----	----	----	----	----



# バブルソート

手順： 1と2を繰り返す  
1. 隣り合う要素を比較  
2. 左 > 右なら交換

初期



1. そのまま
2. 入れ替え
3. 入れ替え

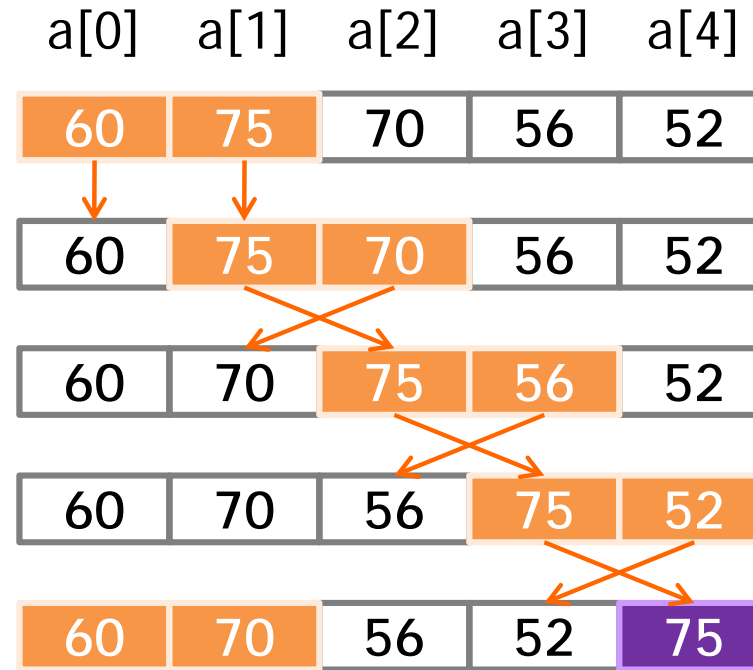
最終



# バブルソート

手順： 1と2を繰り返す  
1. 隣り合う要素を比較  
2. 左 > 右なら交換

初期



a[4]決定 →

1. そのまま
2. 入れ替え
3. 入れ替え
4. 入れ替え

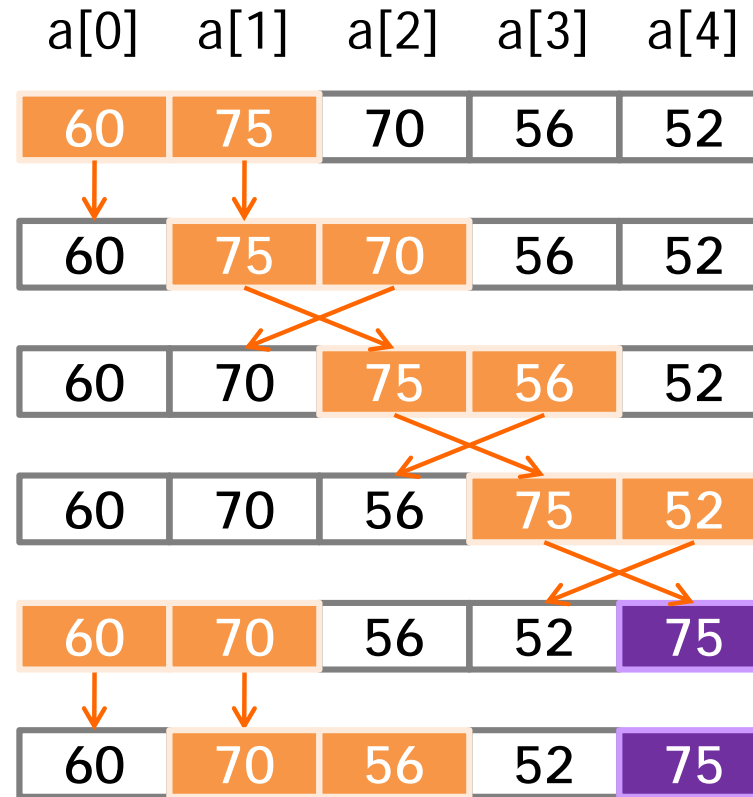
最終

52	56	60	70	75
----	----	----	----	----

# バブルソート

手順： 1と2を繰り返す  
 1. 隣り合う要素を比較  
 2. 左 > 右なら交換

初期



a[4]決定 →

1. そのまま
2. 入れ替え
3. 入れ替え
4. 入れ替え
5. そのまま

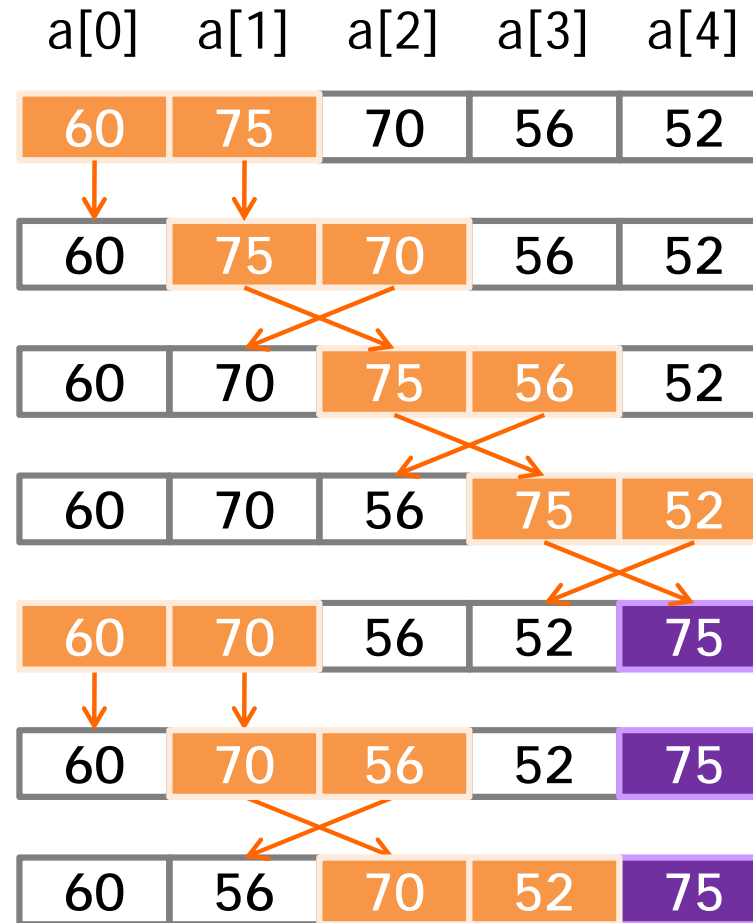
最終

52	56	60	70	75
----	----	----	----	----

# バブルソート

手順： 1と2を繰り返す  
 1. 隣り合う要素を比較  
 2. 左 > 右なら交換

初期



1. そのまま
2. 入れ替え
3. 入れ替え
4. 入れ替え
5. そのまま
6. 入れ替え

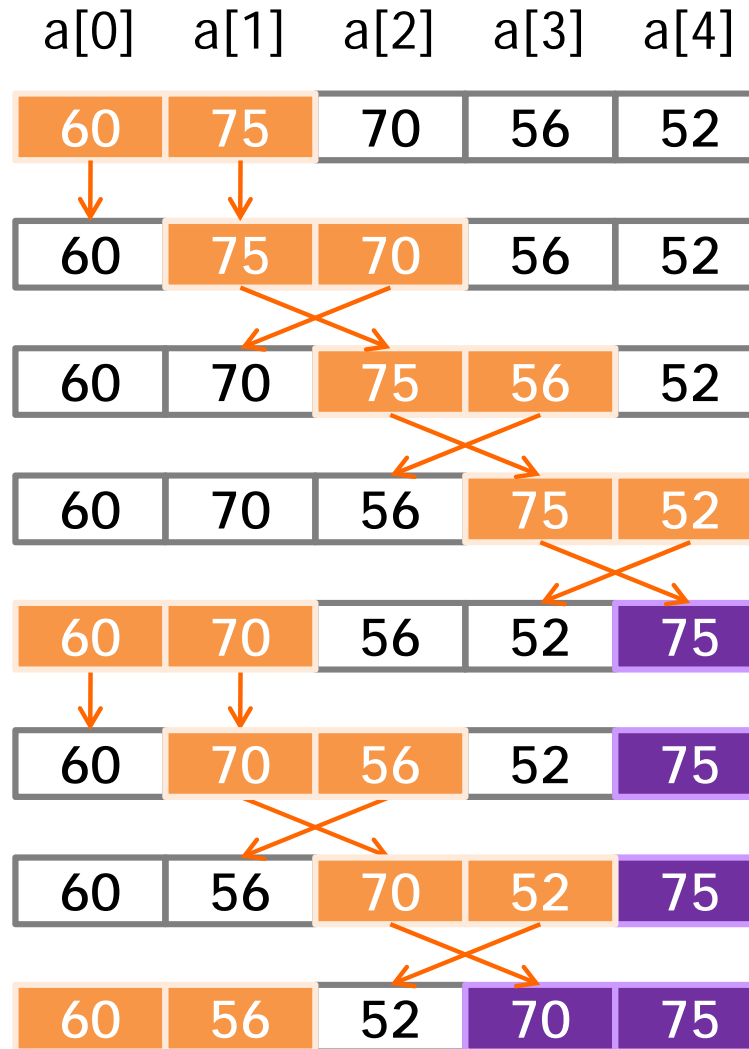
最終



# バブルソート

手順： 1と2を繰り返す  
 1. 隣り合う要素を比較  
 2. 左 > 右なら交換

初期



a[4]決定 →

a[3]決定 →

最終

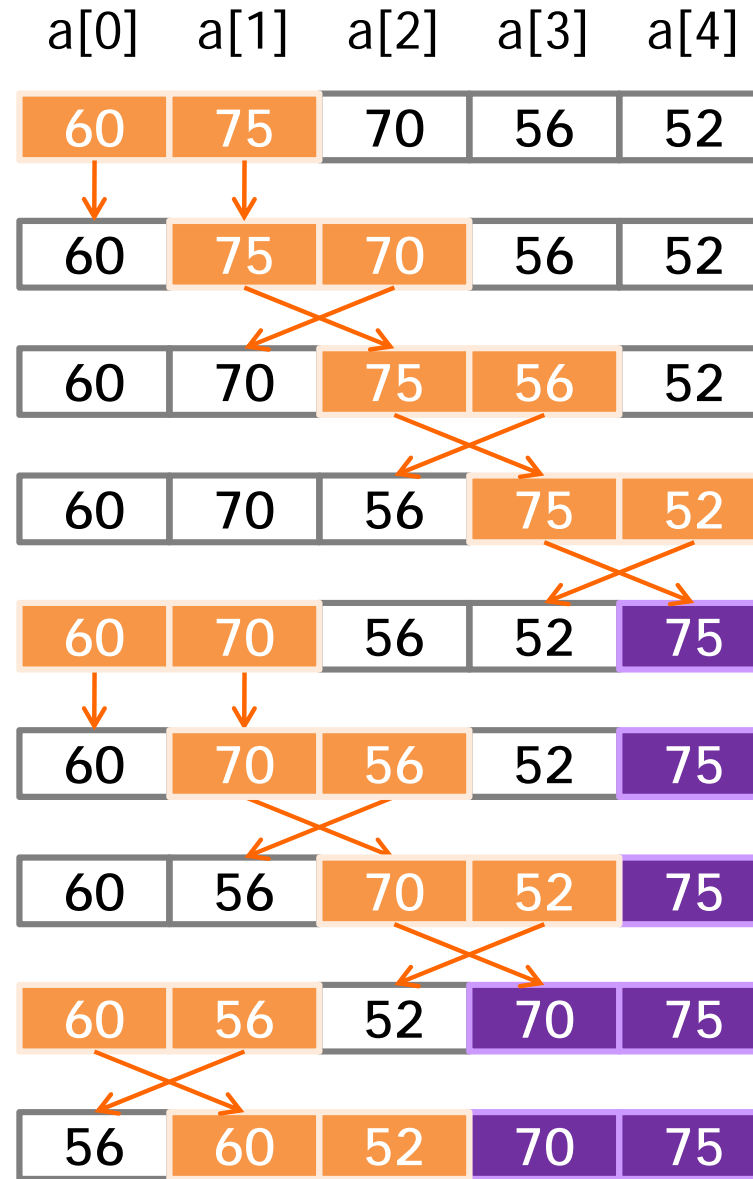
52	56	60	70	75
----	----	----	----	----

1. そのまま
2. 入れ替え
3. 入れ替え
4. 入れ替え
5. そのまま
6. 入れ替え
7. 入れ替え

# バブルソート

手順： 1と2を繰り返す  
 1. 隣り合う要素を比較  
 2. 左 > 右なら交換

初期



a[4]決定 →

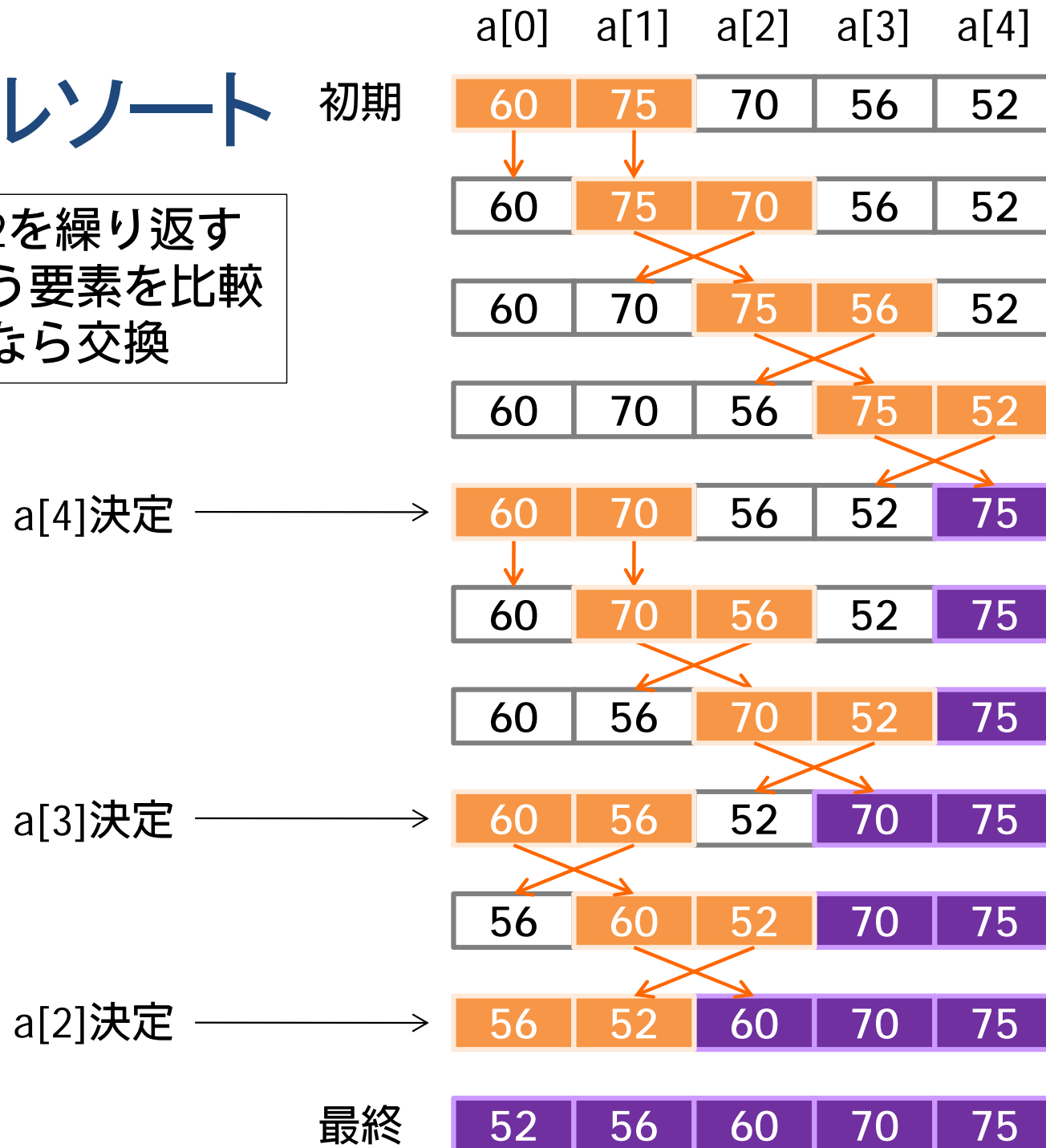
a[3]決定 →

最終

1. そのまま
2. 入れ替え
3. 入れ替え
4. 入れ替え
5. そのまま
6. 入れ替え
7. 入れ替え
8. 入れ替え

# バブルソート

手順： 1と2を繰り返す  
 1. 隣り合う要素を比較  
 2. 左 > 右なら交換

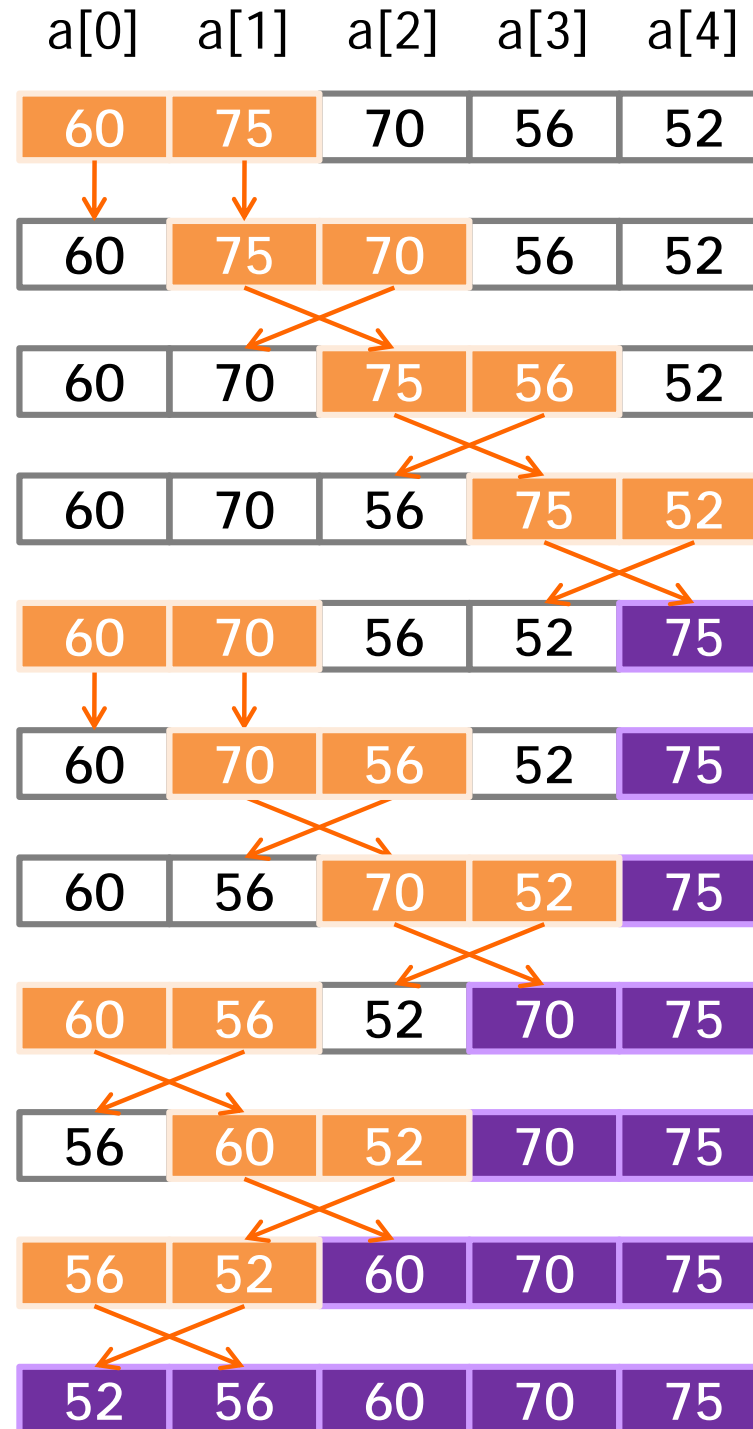


1. そのまま
2. 入れ替え
3. 入れ替え
4. 入れ替え
5. そのまま
6. 入れ替え
7. 入れ替え
8. 入れ替え
9. 入れ替え

# バブルソート

手順： 1と2を繰り返す  
 1. 隣り合う要素を比較  
 2. 左 > 右なら交換

初期



1. そのまま
2. 入れ替え
3. 入れ替え
4. 入れ替え
5. そのまま
6. 入れ替え
7. 入れ替え
8. 入れ替え
9. 入れ替え
10. 入れ替え

a[0], a[1]決定 → 最終



# バブルソートプログラム

```
#include <stdio.h>
```

```
#define NUM 5
```

```
void swap( int *a, int *b )
{
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}
```

```
// バブルソート
```

```
int main(void)
{
    int i, j, k, count ;

    // データ入力
    int a[5] = { 60, 75, 70, 56, 52 };

    printf(“ 整列前データ:”);
    for( i=0;i<NUM;i++){
        printf( “%4d”, a[i] );
    }
}
```

```
// バブルソート
```

```
count = 0;
for( i=NUM-1;i>=0;i-- ){
    for( j=0;j<i;j++ ){
```

```
// 比較回数カウント
```

```
count ++;
```

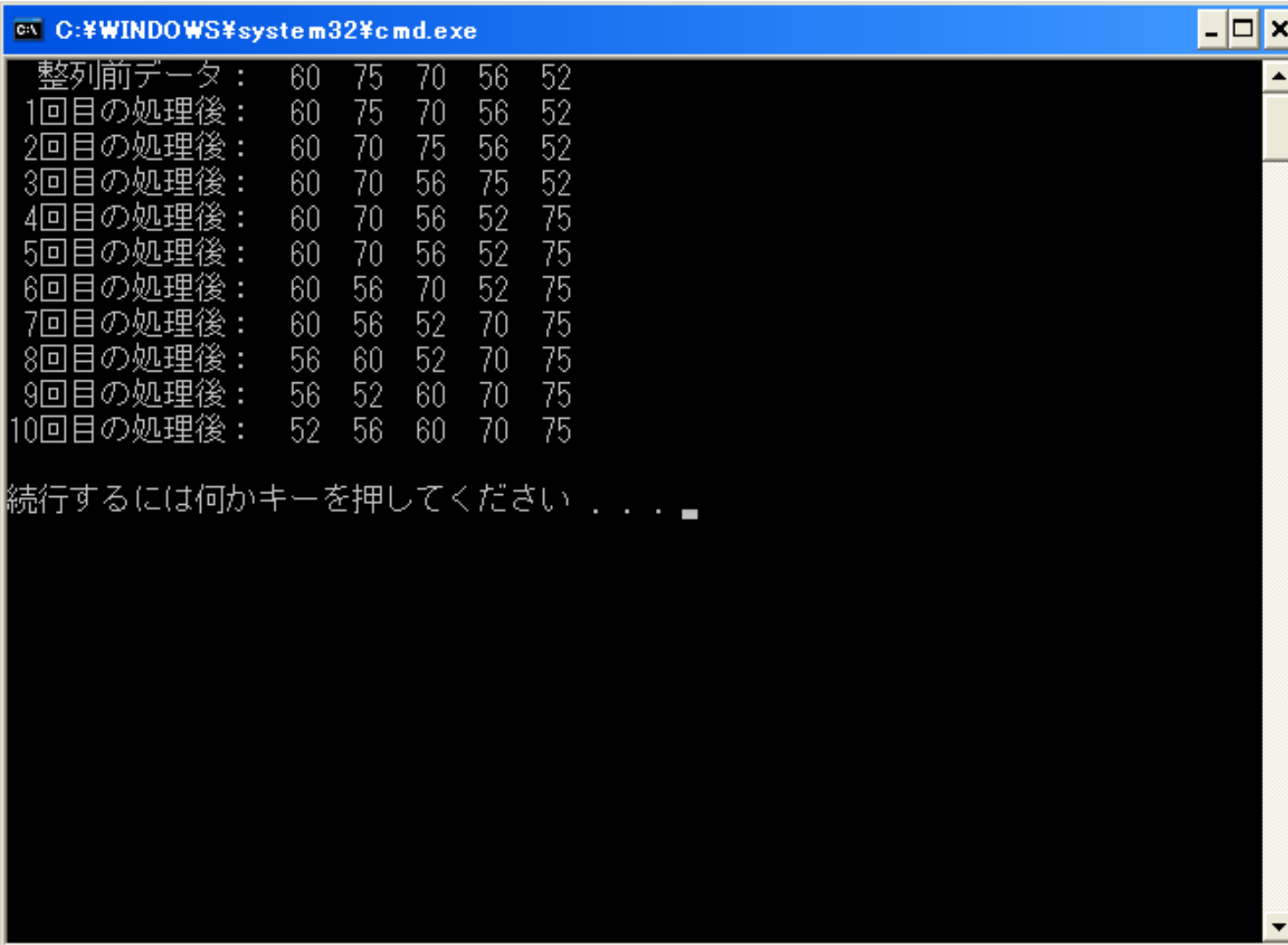
```
// 条件を満たすならば交換
```

```
if( a[j] > a[j+1] ){
    swap( &a[j], &a[j+1] );
}
```

```
// 検証のための表示
```

```
printf( “%2d回目の処理後:”, count
);
for( k=0;k<NUM;k++){
    printf( “%4d”, a[k] );
}
}
```

# バブルソート実行結果



```
C:\WINDOWS\system32\cmd.exe
整列前データ: 60 75 70 56 52
1回目の処理後: 60 75 70 56 52
2回目の処理後: 60 70 75 56 52
3回目の処理後: 60 70 56 75 52
4回目の処理後: 60 70 56 52 75
5回目の処理後: 60 70 56 52 75
6回目の処理後: 60 56 70 52 75
7回目の処理後: 60 56 52 70 75
8回目の処理後: 56 60 52 70 75
9回目の処理後: 56 52 60 70 75
10回目の処理後: 52 56 60 70 75

続行するには何かキーを押してください . . .
```

# 挿入法

- 挿入法とは？
  - 整列してある配列に, 追加要素を適切な場所に挿入することでデータの整列を行う方法
  - バブルソートよりは高度であるが, 普通の方法

# 挿入法

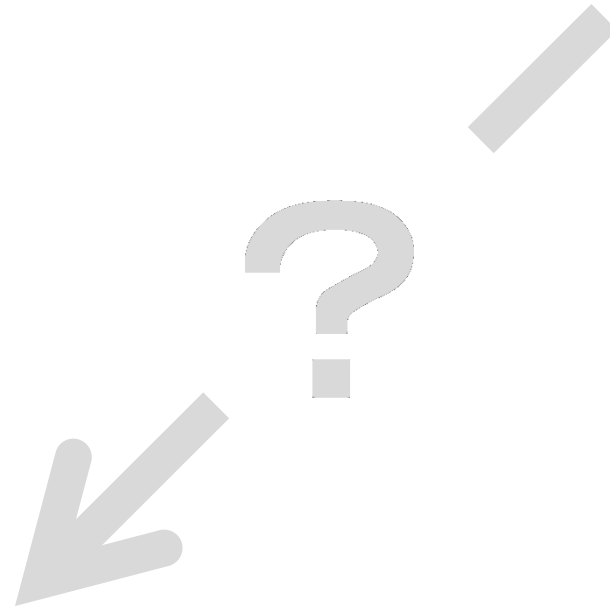
整列後データ

整列前データ

2 6 1 9 5

?

1 2 5 6 9



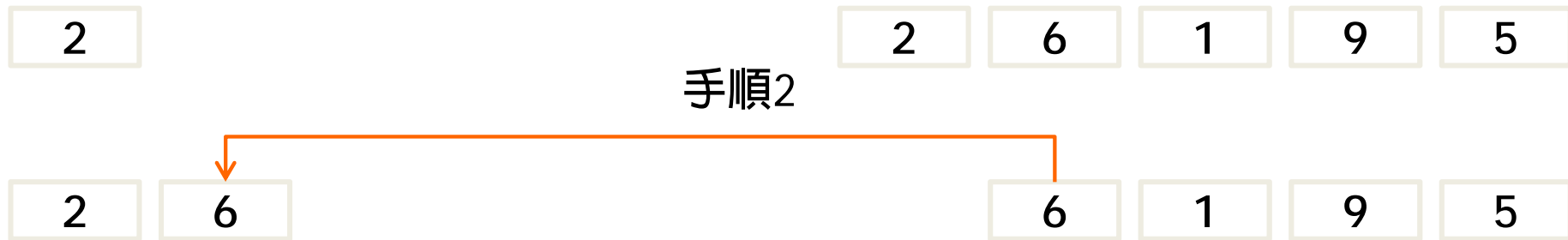
# 挿入法



# 挿入法

整列後データ

整列前データ



# 挿入法

整列後データ

整列前データ

2

2

6

1

9

5

2

6

6

1

9

5

手順3

1

2

6

1

9

5



# 挿入法

整列後データ

整列前データ

2

2

6

1

9

5

2

6

6

1

9

5

1

2

6

1

9

5

手順4

1

2

6

9

9

5





# 挿入法

整列後データ

2

2 6

1 2 6

1 2 6 9

1 2 5 6 9

整列前データ

2 6 1 9 5

6 1 9 5

1 9 5

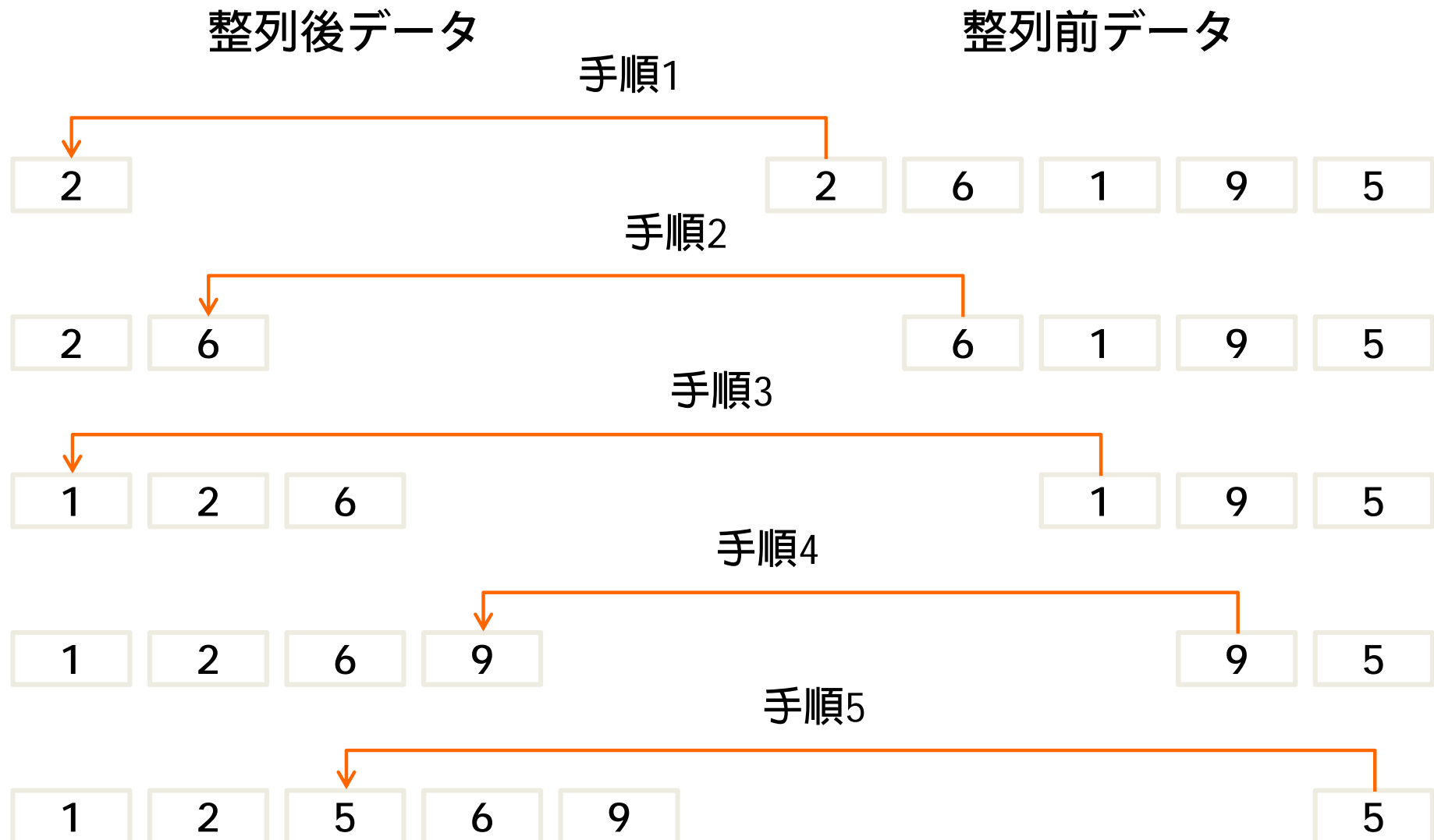
9 5

手順5

5



# 挿入法

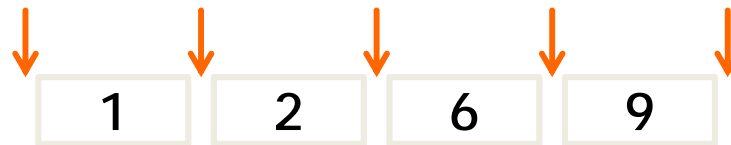


# 挿入法

- 挿入できる場所は5か所

挿入するデータ

5



# 挿入法



$9 > 5$  間違い

# 挿入法



$9 > 5$  間違い

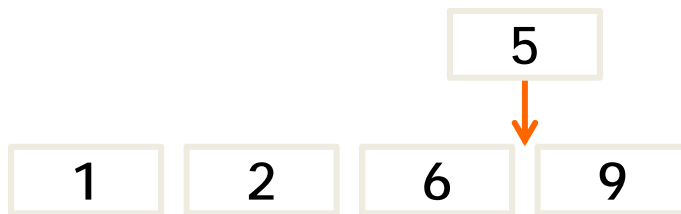


$6 > 5$  間違い

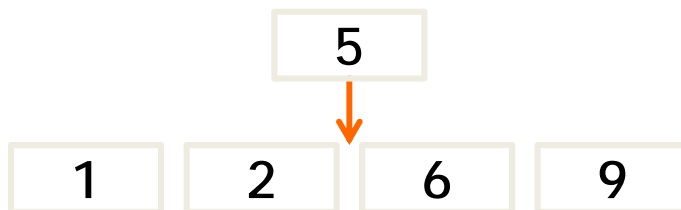
# 挿入法



$9 > 5$  間違い

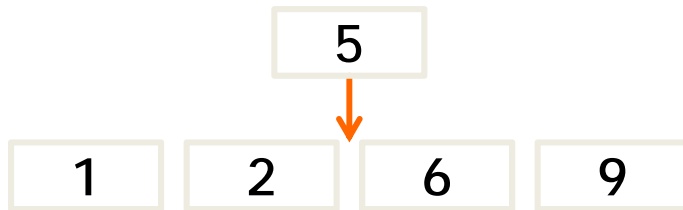


$6 > 5$  間違い

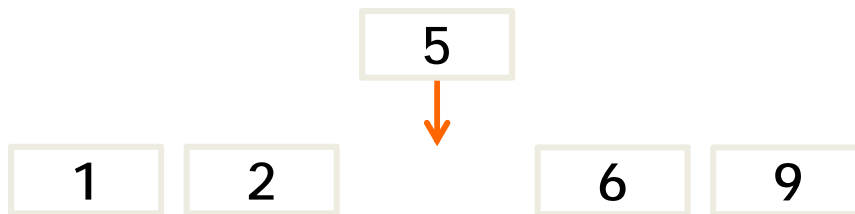


$2 < 5$  正しい

# 挿入法



挿入場所決定

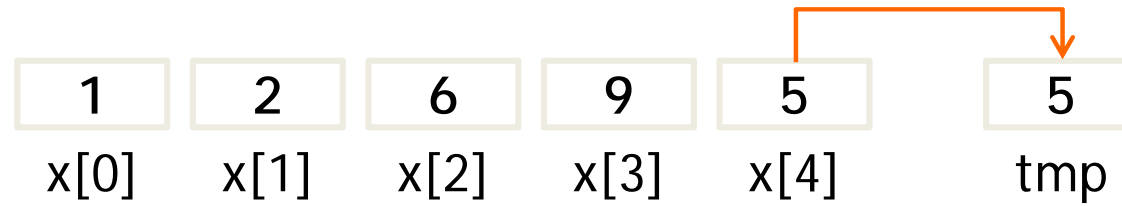


右にずらす  
(スペースを作る)



挿入する

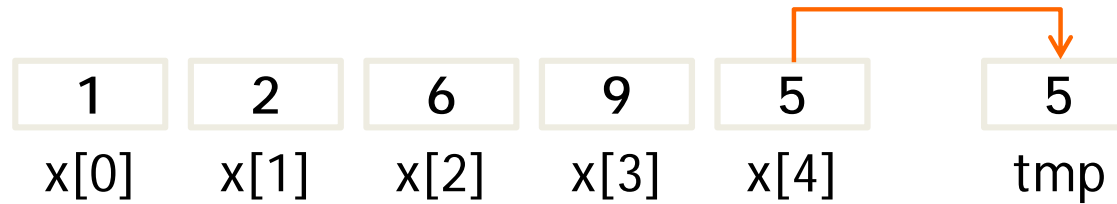
# 挿入法



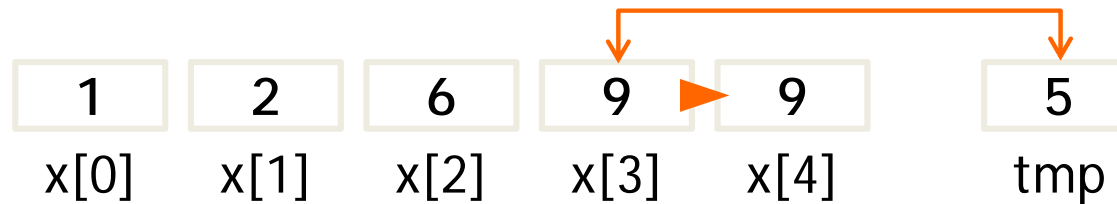
挿入する値を保存



# 挿入法

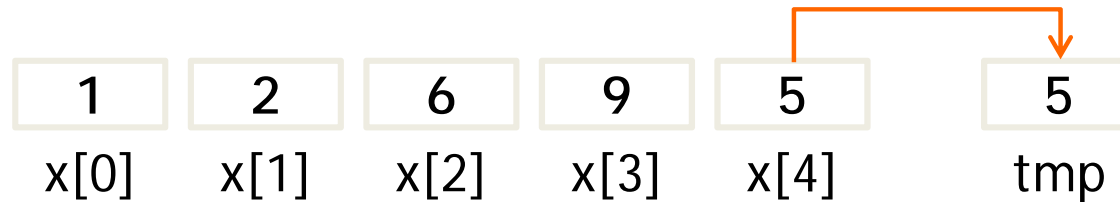


挿入する値を保存

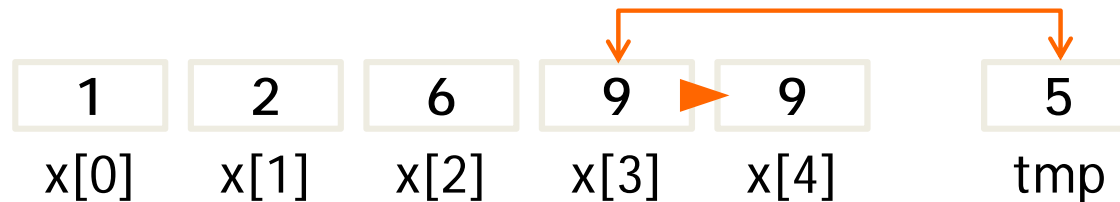


9 > 5なので  
 $x[3]$ を右にずらす

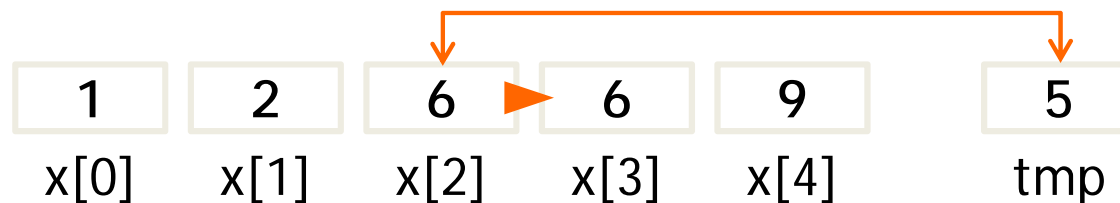
# 挿入法



挿入する値を保存

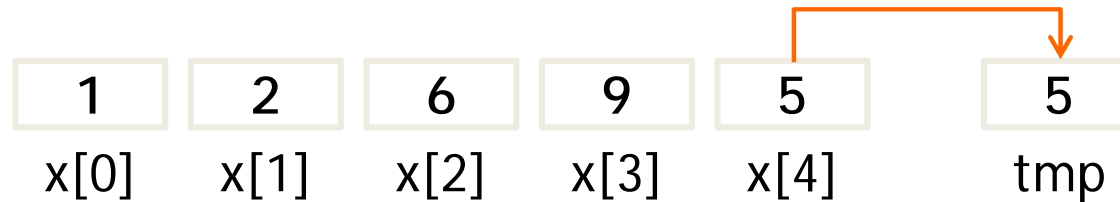


9 > 5なので  
x[3]を右にずらす

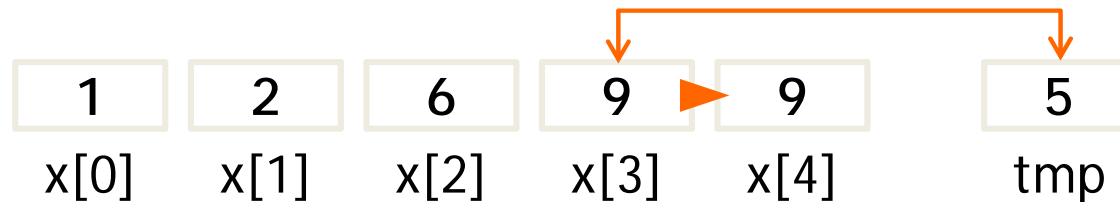


6 > 5なので  
x[2]を右にずらす

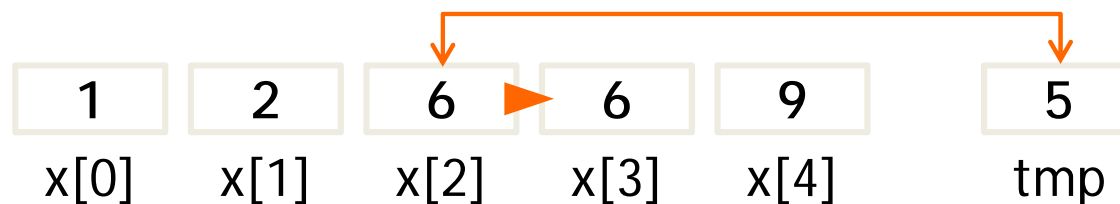
# 挿入法



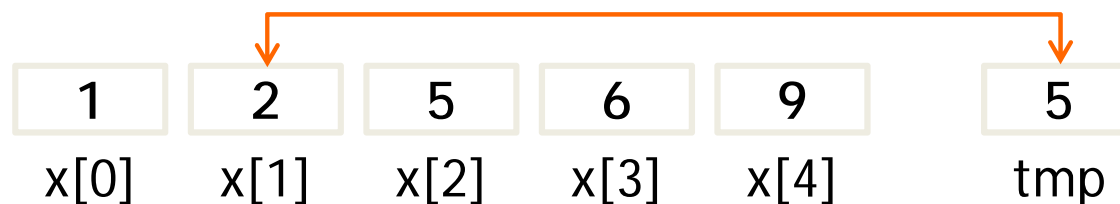
挿入する値を保存



9 > 5なので  
x[3]を右にずらす



6 > 5なので  
x[2]を右にずらす



2 < 5なので  
x[2]に挿入

x[0]

x[1]

x[2]

x[3]

x[4]

tmp

初期

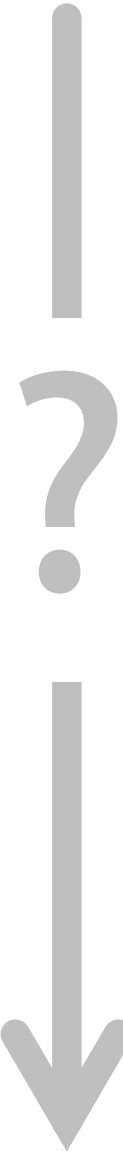
5

2

1

4

3



最終

1

2

3

4

5

		x[0]	x[1]	x[2]	x[3]	x[4]	tmp
初期		5	2	1	4	3	
1	x[0]はそのまま	5	2	1	4	3	

		x[0]	x[1]	x[2]	x[3]	x[4]	tmp
初期		5	2	1	4	3	
1	x[0]はそのまま	5	2	1	4	3	
2	x[1]をtmpに保存	5	2	1	4	3	2
2	挿入候補は2か所	5	2	1	4	3	2
2	x[0]と比較	5	2	1	4	3	2
2	tmp<x[0]なので、右にずらす	5	5	1	4	3	2
2	x[0]に挿入	2	5	1	4	3	2

		x[0]	x[1]	x[2]	x[3]	x[4]	tmp
初期		5	2	1	4	3	
1	x[0]はそのまま	5	2	1	4	3	
2	x[1]をtmpに保存	5	2	1	4	3	2
2	挿入候補は2か所	5	2	1	4	3	2
2	x[0]と比較	5	2	1	4	3	2
2	tmp<x[0]なので、右にずらす	5	5	1	4	3	2
2	x[0]に挿入	2	5	1	4	3	2
3	x[2]をtmpに保存	2	5	1	4	3	1
3	挿入候補は3か所	2	5	1	4	3	1
3	x[1]と比較	2	5	1	4	3	1
3	tmp<x[1]なので、右にずらす	2	5	5	4	3	1
3	x[0]と比較	2	5	5	4	3	1
3	tmp<x[0]なので、右にずらす	2	2	5	4	3	1
3	x[0]に挿入	1	2	5	4	3	1

		x[0]	x[1]	x[2]	x[3]	x[4]	tmp
4	x[3]をtmpに保存	1	2	5	4	3	4
4	挿入候補は4か所	1	2	5	4	3	4
4	x[2]と比較	1	2	5	4	3	4
4	tmp<x[2]なので、右にずらす	1	2	5	5	3	4
4	x[1]と比較	1	2	5	5	3	4
4	tmp>x[1]なので、x[2]に挿入	1	2	4	5	3	4



			x[0]	x[1]	x[2]	x[3]	x[4]	tmp
4		x[3]をtmpに保存	1	2	5	4	3	4
4		挿入候補は4か所	1	2	5	4	3	4
4		x[2]と比較	1	2	5	4	3	4
4		tmp<x[2]なので、右にずらす	1	2	5	5	3	4
4		x[1]と比較	1	2	5	5	3	4
4		tmp>x[1]なので、x[2]に挿入	1	2	4	5	3	4
<hr/>								
5		x[4]をtmpに保存	1	2	4	5	3	3
5		挿入候補は5か所	1	2	4	5	3	3
5		x[3]と比較	1	2	4	5	3	3
5		tmp<x[3]なので、右にずらす	1	2	4	5	5	3
5		x[2]と比較	1	2	4	5	5	3
5		tmp<x[2]なので、右にずらす	1	2	4	4	5	3
5		x[1]と比較	1	2	4	4	5	3
5		tmp>x[1]なので、x[2]に挿入	1	2	3	4	5	3
<hr/>								
最終			1	2	3	4	5	

# 5を挿入する場合の例

iは挿入する要素の番号  
2番目のa[1]から始める

$i=1; i < n; i++$

右から順に

$tmp \leftarrow a[i]$

左の要素と  
値を比較

$j=i-1; j \geq 0; j--$

$a[j] > tmp$ なら  
右にずらす

$a[j] > tmp ?$

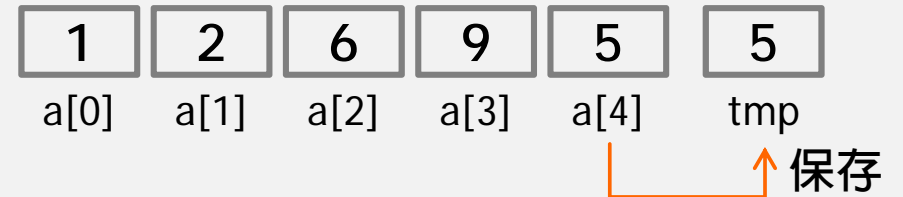
Yes

$a[j+1] = a[j]$

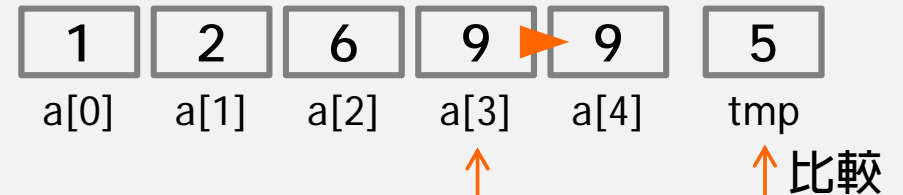
tmpを  
 $a[j+1]$ に挿入

$a[j+1] \leftarrow tmp$

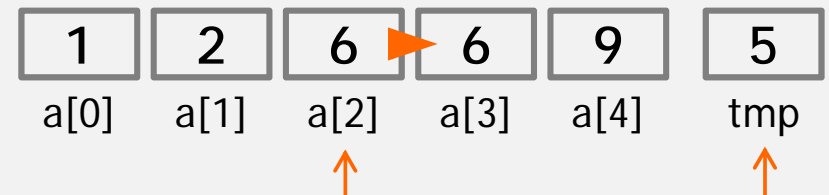
挿入する値をtmpに保存



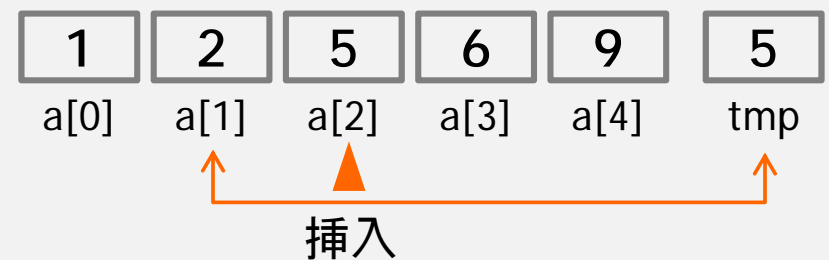
9>5なので、a[3]を右にずらす( $a[4]=a[3]$ )  
(→挿入する場所を確保する)



6>5なので、a[2]を右にずらす( $a[3]=a[2]$ )  
(→挿入する場所を確保する)



2<5なので、a[2]に5を挿入( $a[2]=tmp$ )



# 挿入法プログラム

```
#include <stdio.h>
```

```
#define NUM 5
```

```
// 挿入法ソート
```

```
int main(void)
```

```
{  
    int i, j, k, tmp;
```

```
    // データ入力
```

```
    int a[5] = { 60, 75, 70, 56, 52 };
```

```
    printf("  整列前データ:");
```

```
    for( i=0;i<NUM;i++){
```

```
        printf( "%4d", a[i] );
```

```
    }
```

```
// 挿入法ソート
```

```
for( i=1;i<NUM;i++){
```

```
    // 検証のための表示
```

```
    printf( "%2d回目の処理後：", i );
```

```
    for( k=0;k<i;k++){
```

```
        printf( "%4d", a[k] );
```

```
    }
```

```
        tmp = a[i];
```

```
        for( j=i-1;j>=0;j-- ){
```

```
            if( a[j]>tmp )
```

```
                a[j+1] = a[j];
```

```
            else
```

```
                break;
```

```
        }
```

```
        a[j+1] = tmp;
```

```
    }
```

```
}
```

# 挿入法プログラム(比較, シフト, 挿入回数のカウント)

```

#include <stdio.h>

#define NUM 5

// 挿入法ソート
int main(void)
{
    int i, j, k, tmp;
    int n_comp = 0;
    int n_shift = 0;
    int n_insert = 0;

    // データ入力
    int a[5] = { 60, 75, 70, 56, 52 };

    printf("  整列前データ:");
    for( i=0;i<NUM;i++){
        printf( "%4d", a[i] );
    }

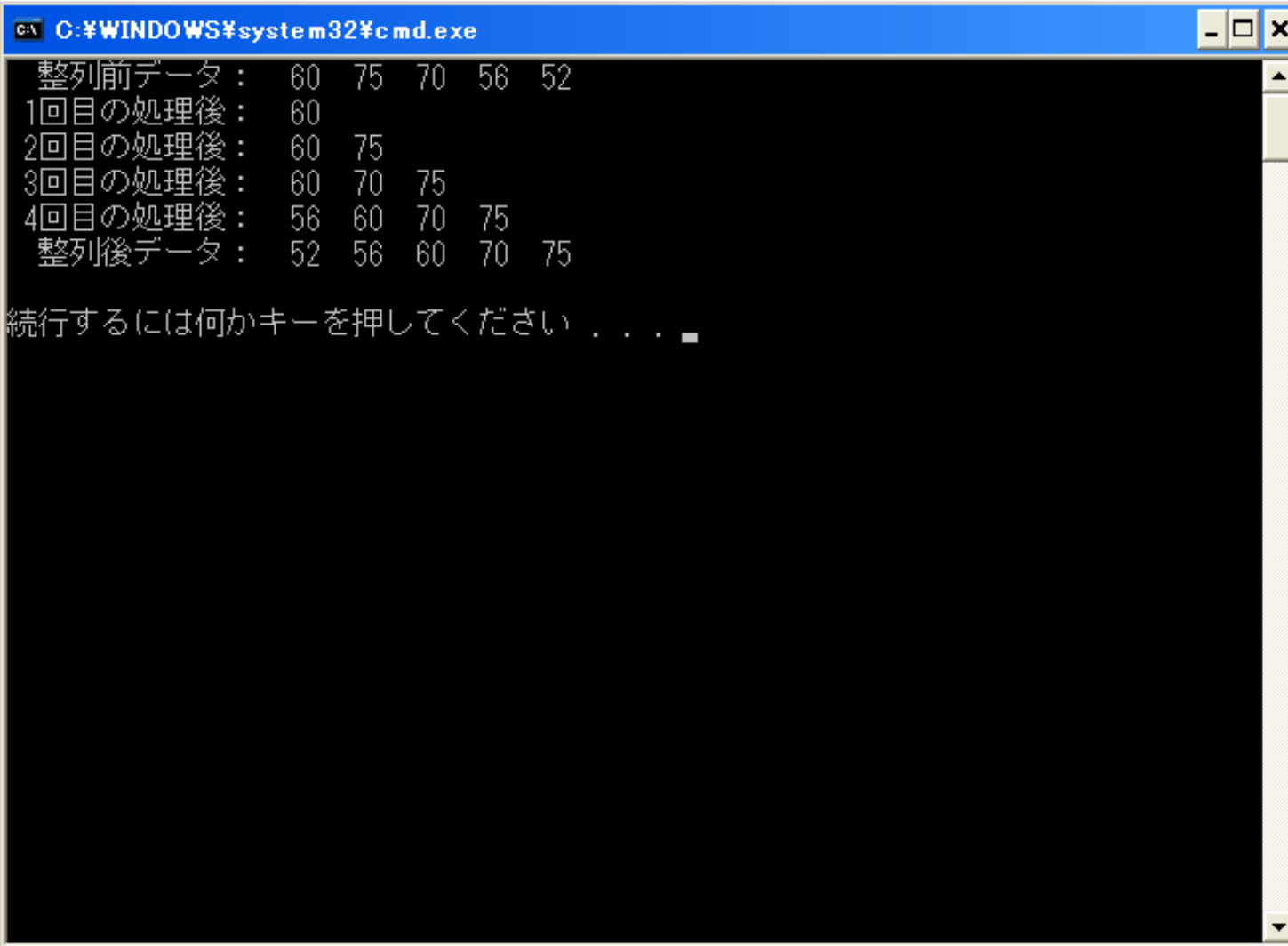
    // 挿入法ソート
    for( i=1;i<NUM;i++){

        // 検証のための表示
        printf( "%2d回目の処理後：", i );
        for( k=0;k<i;k++){
            printf( "%4d", a[k] );
        }

        tmp = a[i];
        for( j=i-1;j>=0;j-- ){
            n_comp++; // 比較回数
            if( a[j]>tmp ){
                n_shift++; // シフト回数
                a[j+1] = a[j];
            }else
                break;
        }
        n_insert++; // 挿入回数
        a[j+1] = tmp;
    }
}

```

# 挿入法実行結果



```
C:\WINDOWS\system32\cmd.exe
整列前データ: 60 75 70 56 52
1回目の処理後: 60
2回目の処理後: 60 75
3回目の処理後: 60 70 75
4回目の処理後: 56 60 70 75
整列後データ: 52 56 60 70 75
続行するには何かキーを押してください . . .
```