第12回 ソート

アルゴリズムとデータ構造ならびに同演習

柏原　昭博

akihiro.kashihara@inf.uec.ac.jp

講義内容

- □　第11回目　ヒープ
- □　**第12回目　ソート（整列）**
- □　第13回目　クイックソート・その他のソート
- □　第14回目　文字列処理（基本）
- □　第15回目　文字列処理（配列処理・文字探索）

ソート（整列）

- データ集合またはデータ系列をキー項目の値の大小関係にしたがって並び替えること
- **順序**
  - **昇順（辞書順）**：小さい順に並べること
    - 1→3→5→7...
    - a→aab→ab→ba→bc→c...
  - **降順**：大きい順に並べること
    - ...→7→5→3→1

ソート（整列）Cont.

- 内部ソートと外部ソート
  - 内部ソート：メモリ上の配列に格納されたデータに対するソート
  - 外部ソート：データ格納領域以外に記憶領域を必要とするソート
- 安定したソート
  - 同じキー値を有する項目に対して，ソート前の順序がソート後も維持されるソート

ソート法の分類

	ソート法	計算量
基本形	基本**交換**法（バブル・ソート）	$O(n^2)$
	基本**選択**法（直接選択法）	
	基本**挿入**法	
改良型	クイック・ソート（改良交換法）	$O(n\log_2 n)$
	ヒープ・ソート（改良選択法）	
	シェル・ソート（改良挿入法）	$O(n^{1.2})$

基本交換法：バブル・ソート

- 昇順に並び替える場合
  - 配列の最後尾から隣接する要素間を順に比較し，最小の値を求める
    - a[j-1] > a[j]　ならば，値を交換
      → 小さい値を順に前方に移動させる
  - 同様にして，次に最小の値を求める

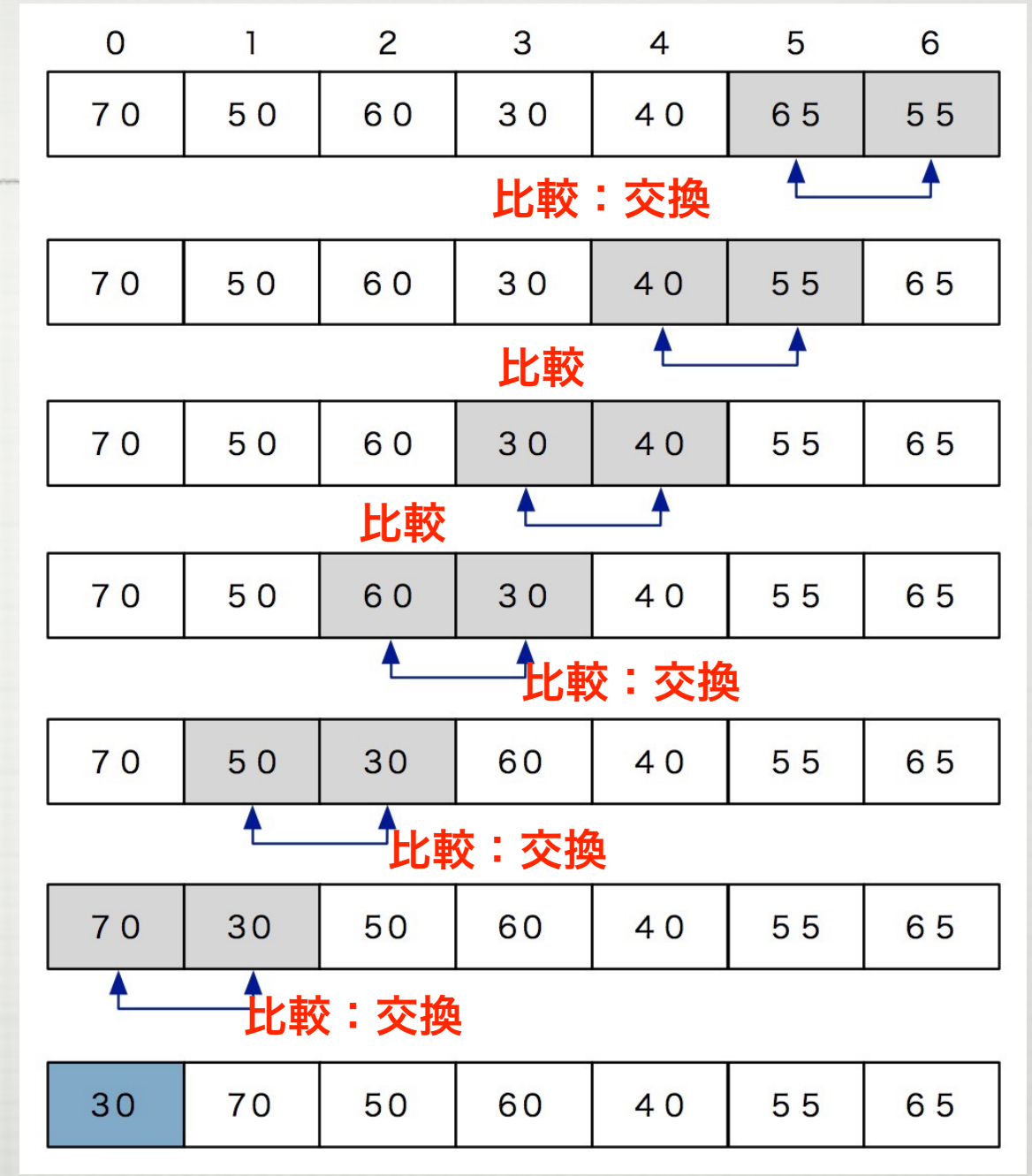
基本交換法
バブル・ソート

n(=7)要素の配列を昇順にソート

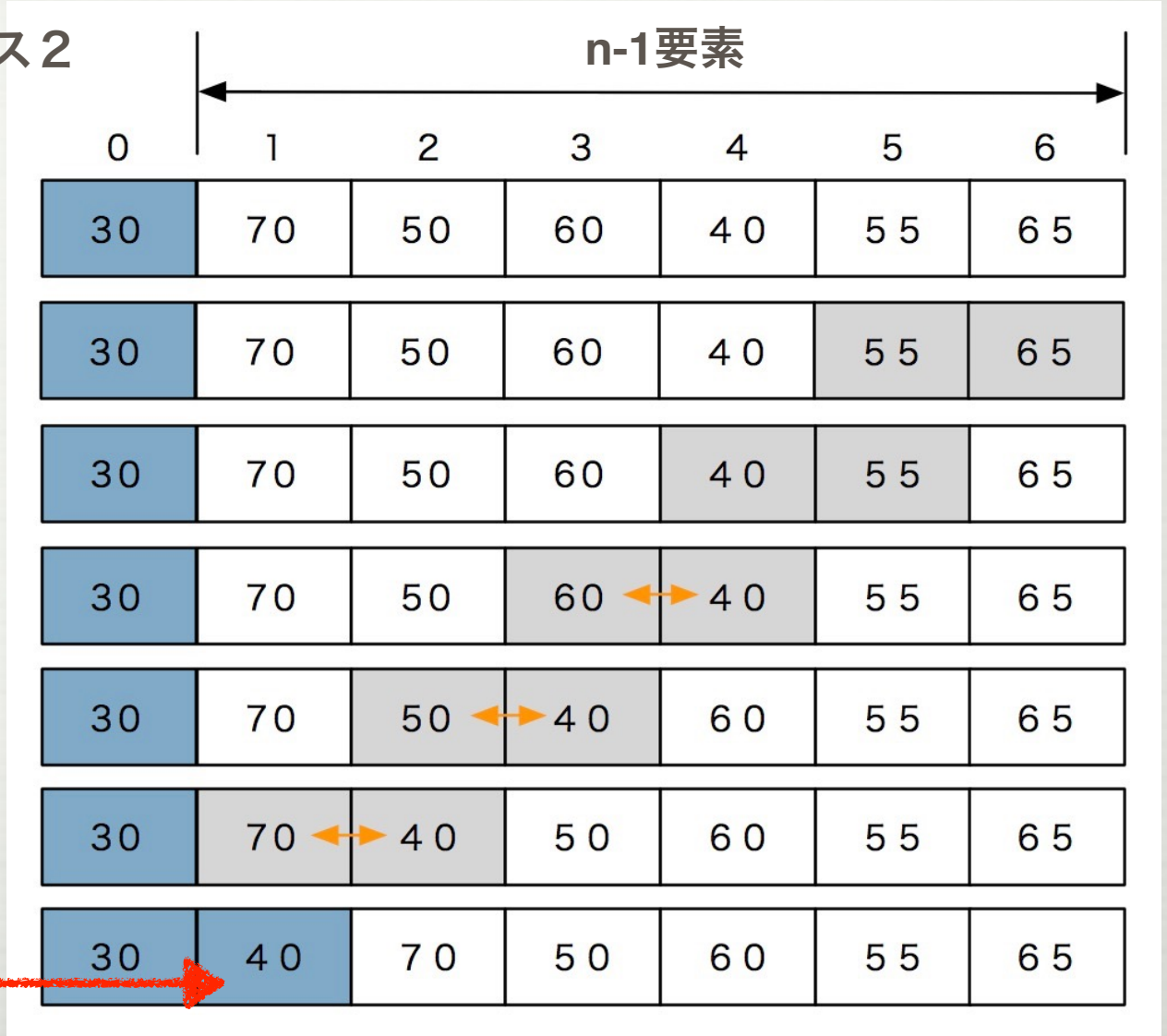
比較回数：n-1回

n個のうちの最小値 →

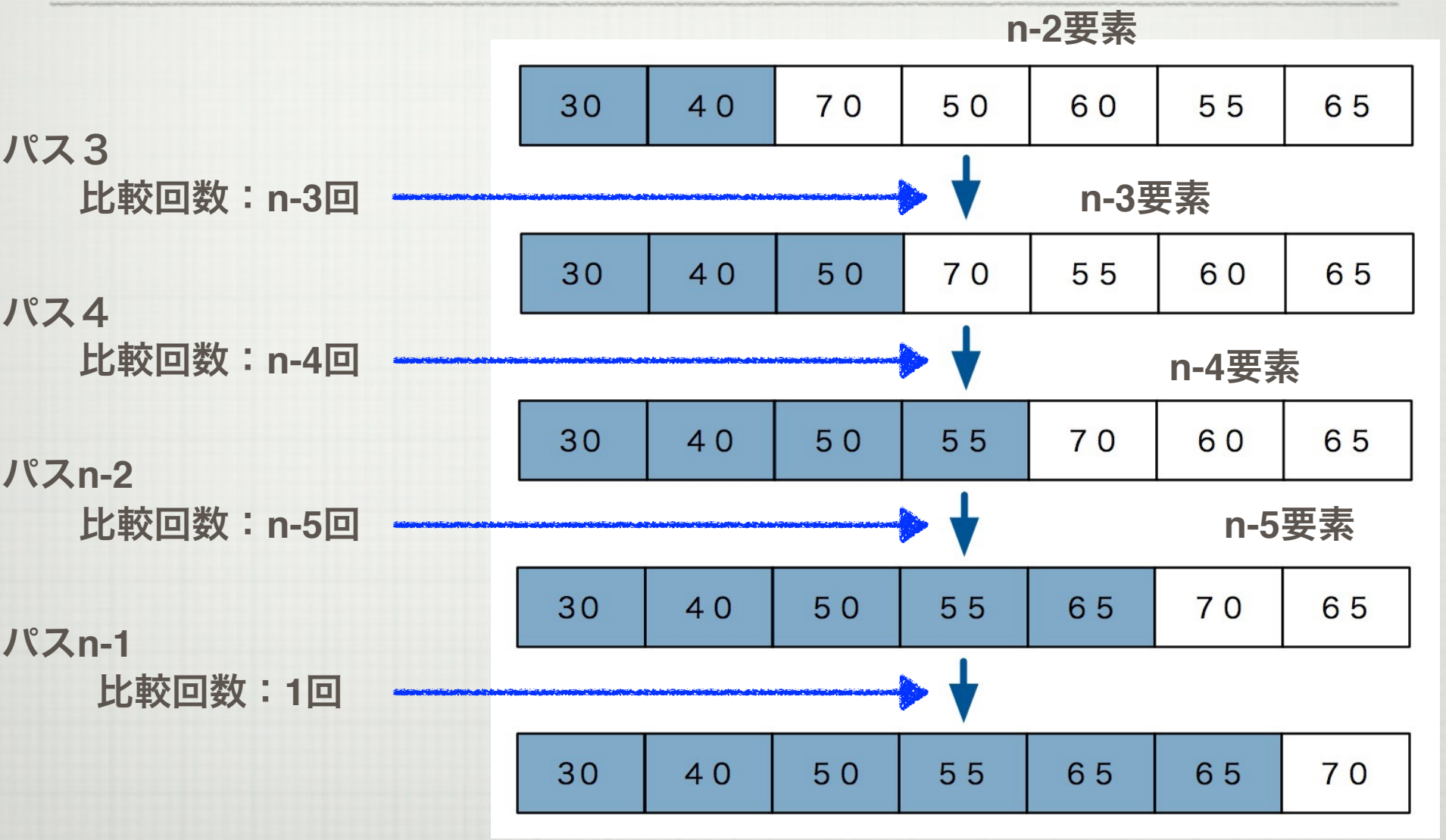
パス１



基本交換法
バブル・ソート



基本交換法
バブル・ソート



バブルソート

```
for (i=0; i < n-1; i++)
    for (j=n-1; j > i; j--)
        if (a[j] < a[j-1])
            a[j]とa[j-1]を交換;
```

...パス1からn-1まで繰り返す

...末尾要素から順に比較・交換

比較回数= (n-1)+(n-2)+(n-3)+...+2+1= n(n-1)/2

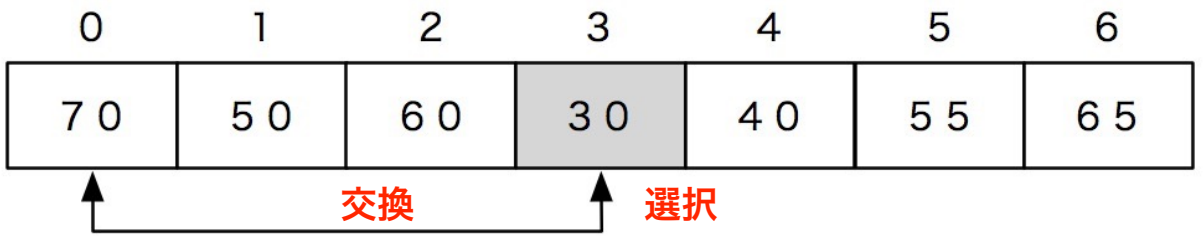
基本（直接）選択法

- 昇順に並び替える場合
  - n要素（未整列の列）のうち，最小の値を選択する
- 配列の先頭要素とその最小値を交換する
  - 先頭要素を除く(n-1)要素からなる未整列の列に対して，同様に最小値を選択し，その配列の先頭要素と交換する.

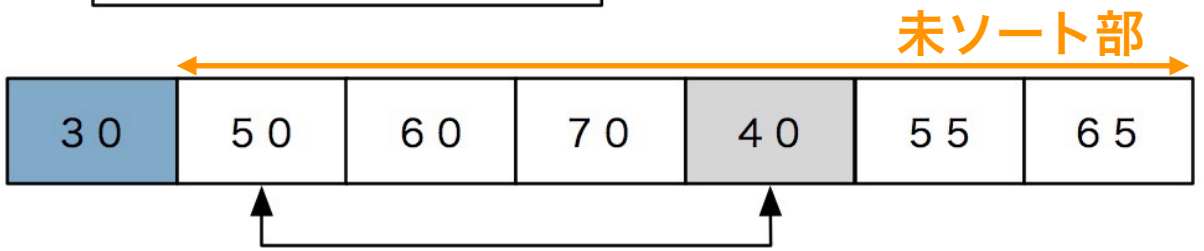
基本（直接）選択法

比較回数：n-1回

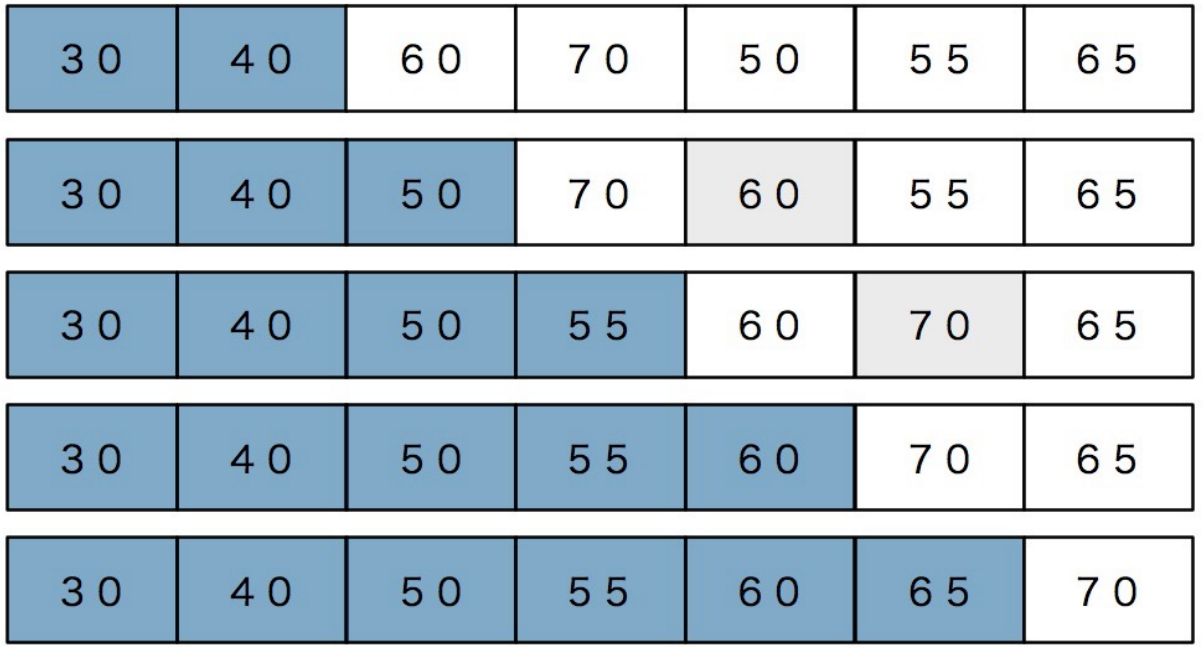
n要素のうち最小値を探し（選択），先頭要素と交換



未ソート部分（n-1要素）のうち最小値を探し，未ソート部分の先頭と交換



比較回数：n-2回



比較回数：1回

基本（直接）選択法

```
for (i=0; i < n-1; i++){
    min = i;
    for (j=i+1; j < n; j++)
        if (a[j] < a[min])
            min = j;
    a[i]をa[min]と交換;
}
```

...未ソート部分が無くなるまで

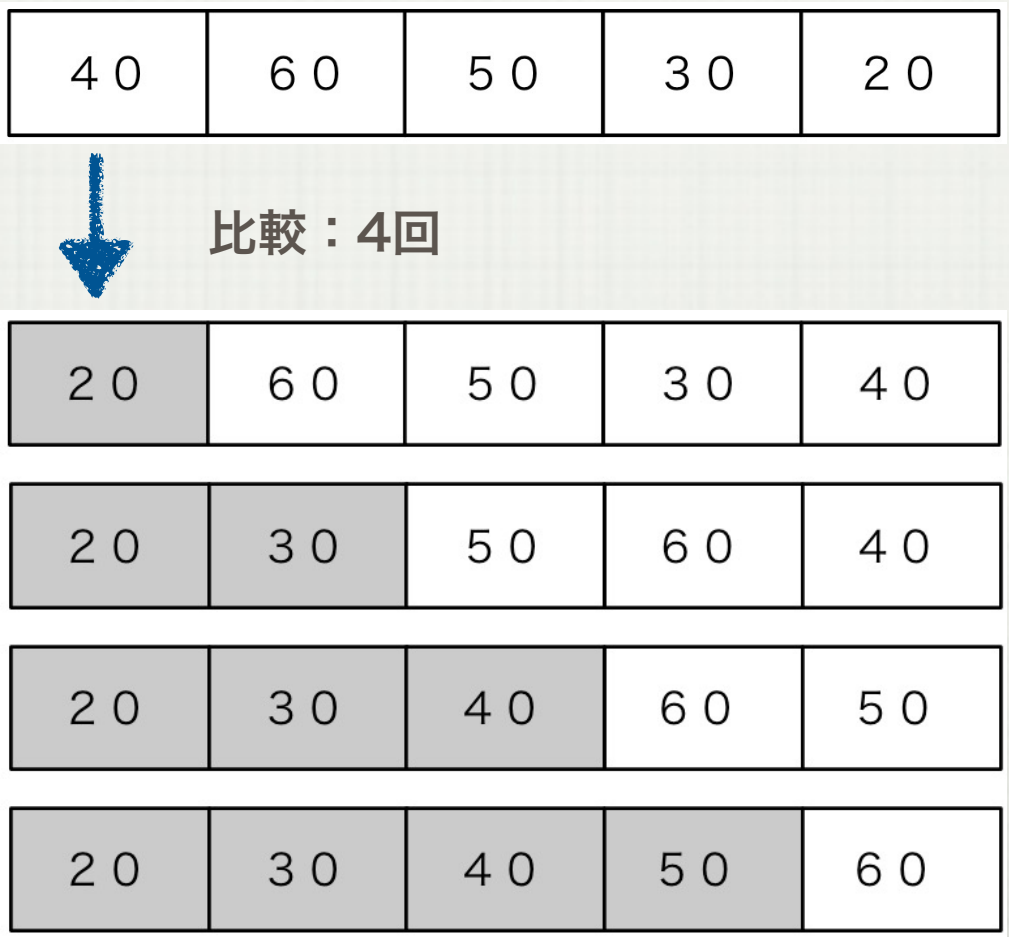
...未ソート部分の最小値を選択

...未ソート部分の先頭要素と交換

比較回数= (n-1)+(n-2)+(n-3)+...+2+1= n(n-1)/2

練習問題12-1

以下のデータ系列を基本選択法で昇順に整列しなさい.
また，整列にかかる比較回数を答えなさい.



答え

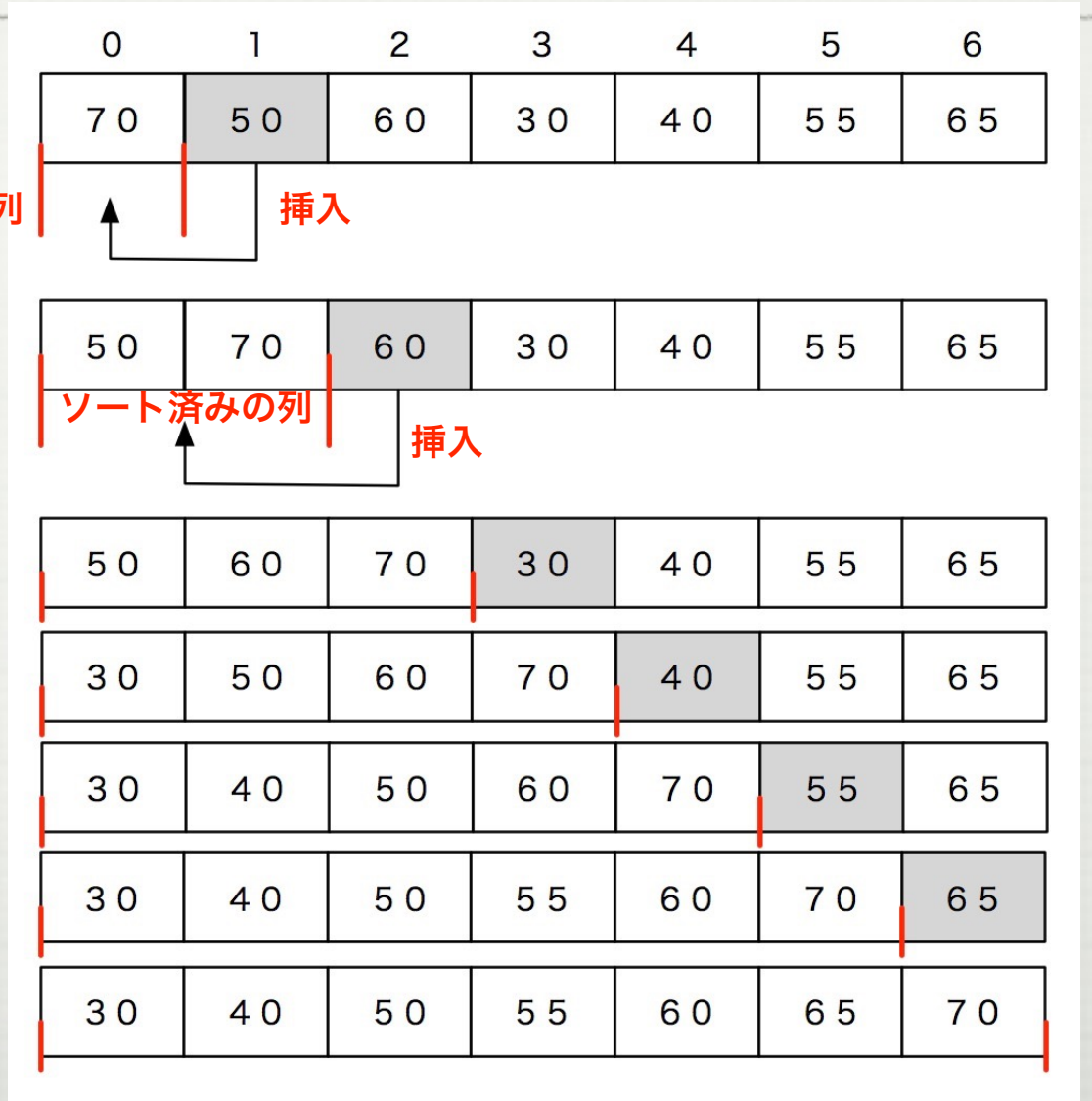
比較：10回
=5(5-1)/2

基本挿入法

- 整列済みの列と未整列の列に分ける
  - 整列済みの列に，未整列の列の先頭要素を挿入する．
    整列済みの列の最後尾から比較し，適切な位置に挿入
    →整列済みの列を順に大きくする方法

基本挿入法

ソート済みの列に，未ソート部分の先頭要素を挿入



基本挿入法

```
for (i=1; i < n; i++){
    tmp=a[i];
    for (j=i-1; j>=0 && a[j]>tmp; j--)
        a[j+1] = a[j];
    a[j+1]=tmp;
}
```

…未ソート部分が無くなるまで

…挿入要素（未ソート部分の先頭要素）

…ソート済みの列に挿入；
挿入要素以下の要素が見つかるまで

…ソート済み要素をずらす

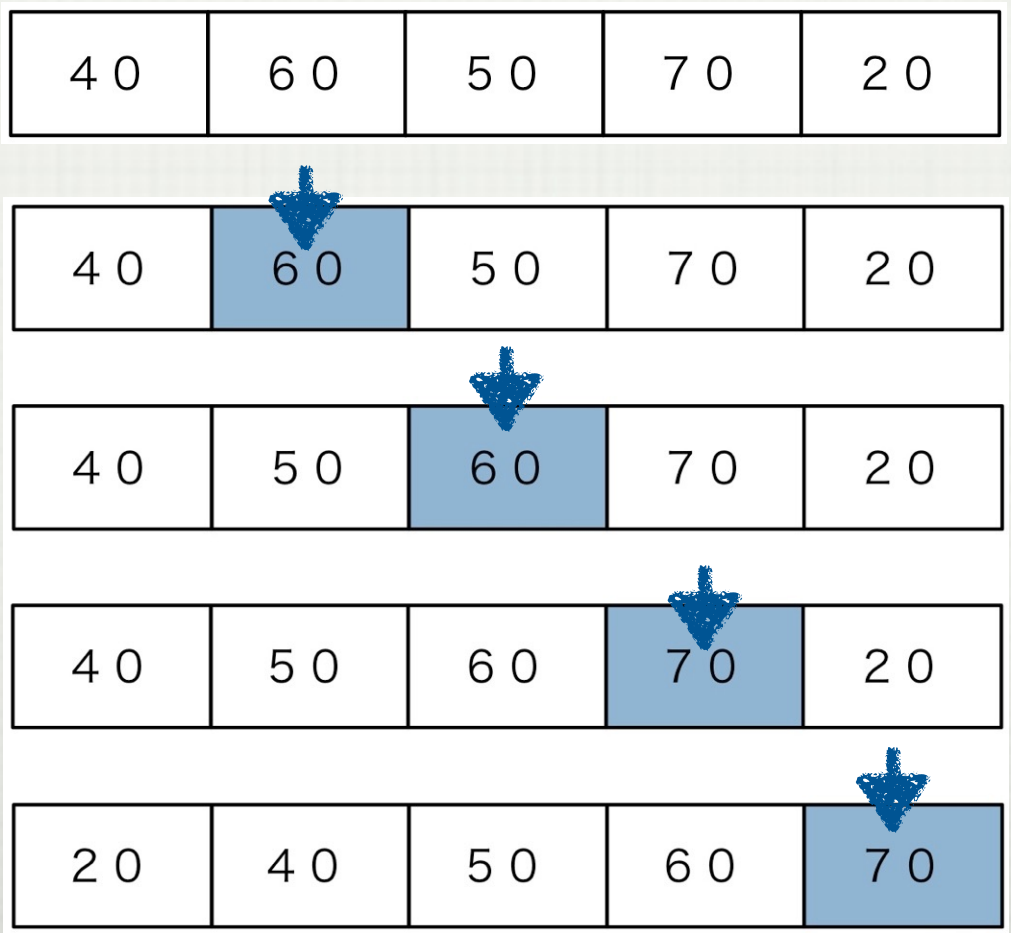
…挿入要素を適切な場所に挿入

練習問題12-2

以下のデータ系列を基本挿入法で昇順に整列しなさい.
また, 整列にかかる比較回数を答えなさい.

答え

比較：8回



練習問題12-3

基本挿入法で昇順に整列する場合，比較回数が最悪（n*(n-1)/2）になる場合はどんな場合か答えなさい.

答え　データ系列が降順になっている場合
例：

７０	６０	５０	４０	３０

演習12-1

- sample12-1.c に，配列要素を基本交換法によって昇順に整列する関数を示す．このプログラムを動作させ，整列するデータ列を色々と変えて，比較・交換回数がどう変化するかを理解しなさい．

  また，このプログラムを参考にして，基本選択法，基本挿入法によって配列要素を昇順に整列する関数（sample12-2.c/12-3.c）を書き，実行しなさい．

sample12-1.c

```c
void bubble_sort (int a[], int n)
{
  int i,j,k;
  int comp=0, exchange=0;
  int tmp;

  for (i=0; i<n-1; i++){ //パス1からn-1まで繰り返す
   for (j=n-1; j>i; j--){
      if (a[j-1]>a[j]){//要素の比較
        tmp=a[j-1];//要素の交換
        a[j-1]=a[j];
        a[j]=tmp;
         exchange++;//交換回数
      }
      comp++;//比較回数
   }
  for (k=0;k<n;k++)
   printf("a[%d]=%d ",k,a[k]);
  printf(" \n");
  }
  printf("Compare:%d   Exchange:%d \n", comp, exchange);
}
```

sample12-2.c

```c
void selection_sort (int a[], int n)
{
  int i,j,k;
  int comp=0, exchange=0;
  int min,tmp;

  for (i=0; i<n-1; i++){//未ソート部分が無くなるまで
   min=i;
   for (j=i+1; j<n; j++){ //未ソート部分の最小値を選択
      if (a[j]<a[min])
         min=j;
      comp++;//比較回数
   }
   tmp=a[i];//最小値と未ソート部分の先頭要素を交換
   a[i]=a[min];
   a[min]=tmp;
   exchange++;//交換回数
   for (k=0;k<n;k++)
      printf("a[%d]=%d ",k,a[k]);
   printf(" \n");
  }
  printf("Compare:%d   Exchange:%d \n", comp, exchange);
}
```

sample12-3.c

```c
void insertion_sort (int a[], int n)
{
  int i,j,k;
  int comp=0, exchange=0;
  int tmp;

  for (i=1; i<n; i++){//未ソート部分が無くなるまで
   tmp=a[i];//挿入要素（未ソート部分の先頭要素）
   for (j=i-1; j>=0 && a[j]>tmp; j--) {//ソート済みの列に挿入
      a[j+1]=a[j];
      comp++; exchange++;//選択回数・交換回数
   }
   a[j+1]=tmp;//挿入要素を適切な場所に挿入
   exchange++;//交換回数
   for (k=0;k<n;k++)
      printf("a[%d]=%d ",k,a[k]);
   printf(" \n");
  }
  printf("Compare:%d   Exchange:%d \n", comp, exchange);
}
```

演習12-2

- 氏名と年齢が対となったデータの集合に対して，氏名の辞書順に整列し，整列した順に氏名および年齢を表示するプログラムを作成しなさい．（sample12-4.c）

  ヒント：構造体で表現されるデータの場合，値の交換に時間を要してしまう．そこで，構造体データを指すポインタ配列を用意し，交換はポインタの交換として行えば良い．

sample12-4.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100

typedef struct {
   char *name;
   int age;
} Members;

void selection_sort(Members *[], int);

int main (void)
{
   Members a[]={"Itoh", 24,//データ集合の定義
    "Ota", 27,
    "Akiyama", 22,
    "Shiota", 29,
    "Murase", 19,
    "Kaneko", 31,
    "Otsuka", 25,
    "Saito", 28};
   int n= sizeof(a)/sizeof(a[0]);//データ数
   int i;
   Members *p[n];
   for (i=0; i<n; i++)//ポインタ配列の準備
    p[i]=&a[i];
   for (i=0; i<n; i++)
    printf("%d: %s  Age=%d \n", i, p[i]->name, p[i]->age);
   printf("\n");
   selection_sort(p, n);//基本選択法でのソート
   return 1;
}
```

sample12-4.c

```c
void selection_sort (Members *p[], int n)
{
  int i, min_i,j,k;
  char *min;
  Members *tmp;

  for (i=0; i<n-1; i++){
   min=p[i]->name;
   min_i=i;
   for (j=i+1; j<n; j++){
      if (strcmp (p[j]->name, min)<0){//未ソート部分の最小値となる名前を選択
         min=p[j]->name;
         min_i=j;
      }
   }
   tmp=p[i];//最小値と未ソート部分の先頭要素を交換
   p[i]=p[min_i];
   p[min_i]=tmp;

   for (k=0;k<n;k++)
      printf("%d: %s  Age=%d \n", k, p[k]->name, p[k]->age);
   printf(" \n");
  }
}
```

課題12-1

- sample12-4.cを参考にして，以下の構造体で表現されるデータの集合に対して，英語・数学・物理の合計点順（降順）に整列し，整列した順に氏名および各科目の点数と合計点を表示するプログラムを作成しなさい.

```
struct Seiseki {
  char *name;
  int eng;
  int math;
  int phy;
  }
```

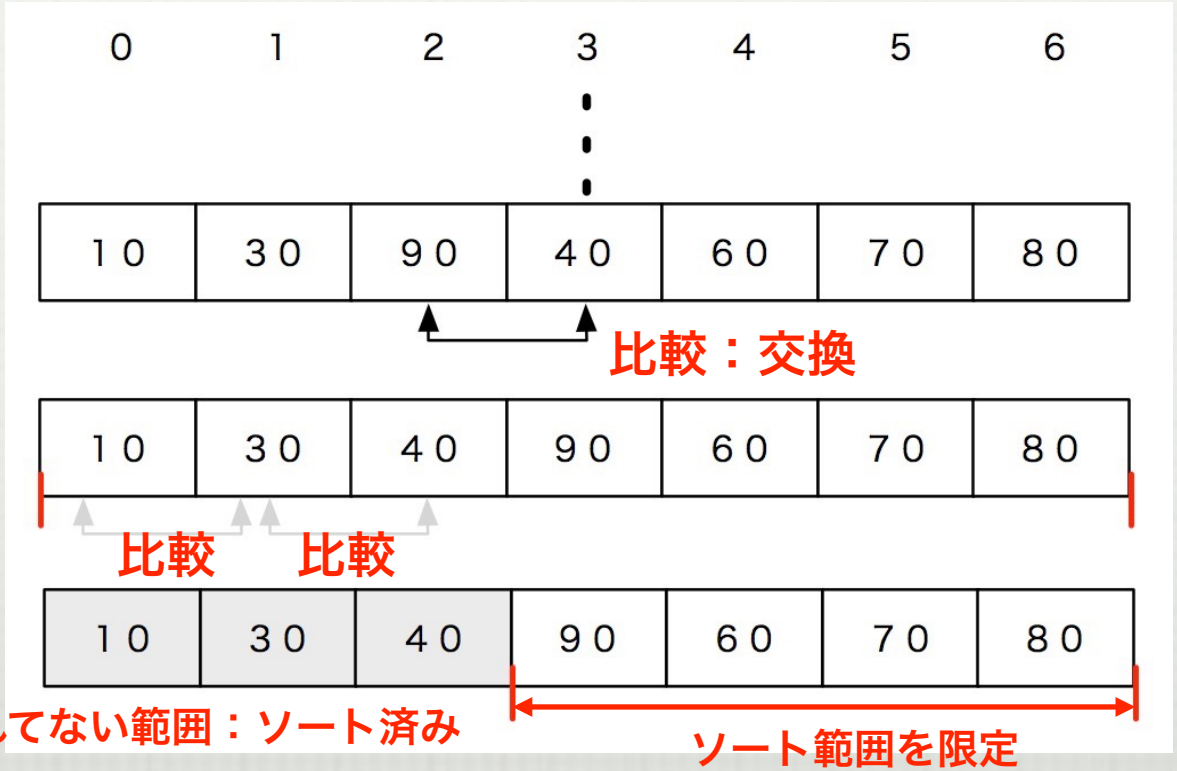
基本形ソート法の改善

- 基本交換法（バブルソート）の改善
  - 比較交換範囲の限定
  - 双方向バブルソート（シェーカーソート）

- 基本挿入法の改善
  - 二分挿入ソート

演習12-3 バブルソートの改善：比較範囲の限定

- sample12-1.c を参考に，比較範囲を限定したバブルソートのプログラムを作成しなさい（sample12-5.c）．また，整列するデータ列を色々と変えて，比較範囲が変化することを確認しなさい．



バブルソートの改善

```
k = 0;
while (k < n-1)
    terminal = n-1;
    for (j=n-1; j > k; j--)
        if (a[j] < a[j-1]){
            a[j]とa[j-1]を交換;
            terminal = j;
        }
    k = terminal;
```

...k:ソート範囲の先頭

...末尾要素から順に比較・交換

...交換があった位置

...最後に交換があった位置

sample12-5.c

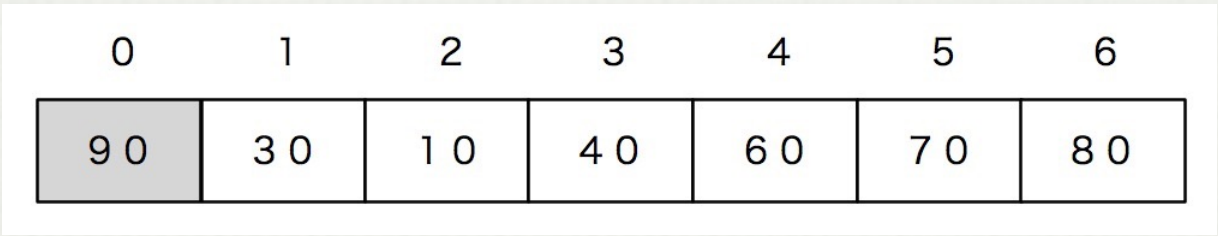
```c
void bubble_sort (int a[], int n)
{
  int j,k,q;
  int comp=0, exchange=0;
  int terminal, tmp;

  k=0;
  while (k<n-1) {//パス1からn-1まで繰り返す
   terminal = n-1;
   for (j=n-1; j>k; j--){//前回最後の交換があった位置まで繰り返す
      if (a[j-1]>a[j]){//要素の比較
         tmp=a[j-1];
         a[j-1]=a[j];
         a[j]=tmp;
         terminal=j;//交換があった位置
         exchange++;//交換回数
      }
      comp++;//比較回数
   }
   k=terminal;//最後に交換があった位置
  }
  printf("Compare:%d   Exchange:%d \n", comp, exchange);
}
```

演習12-4 双方向バブルソート

- sample12-1.c を参考に，双方向バブルソートのプログラムを作成しなさい（sample12-6.c）．そして，先頭に最大値がある場合のバブルソートでの比較・交換回数と双方向バブルソートでの比較交換回数を確かめなさい．

先頭に最大値がある場合，一つずつ後ろへ移動：非効率

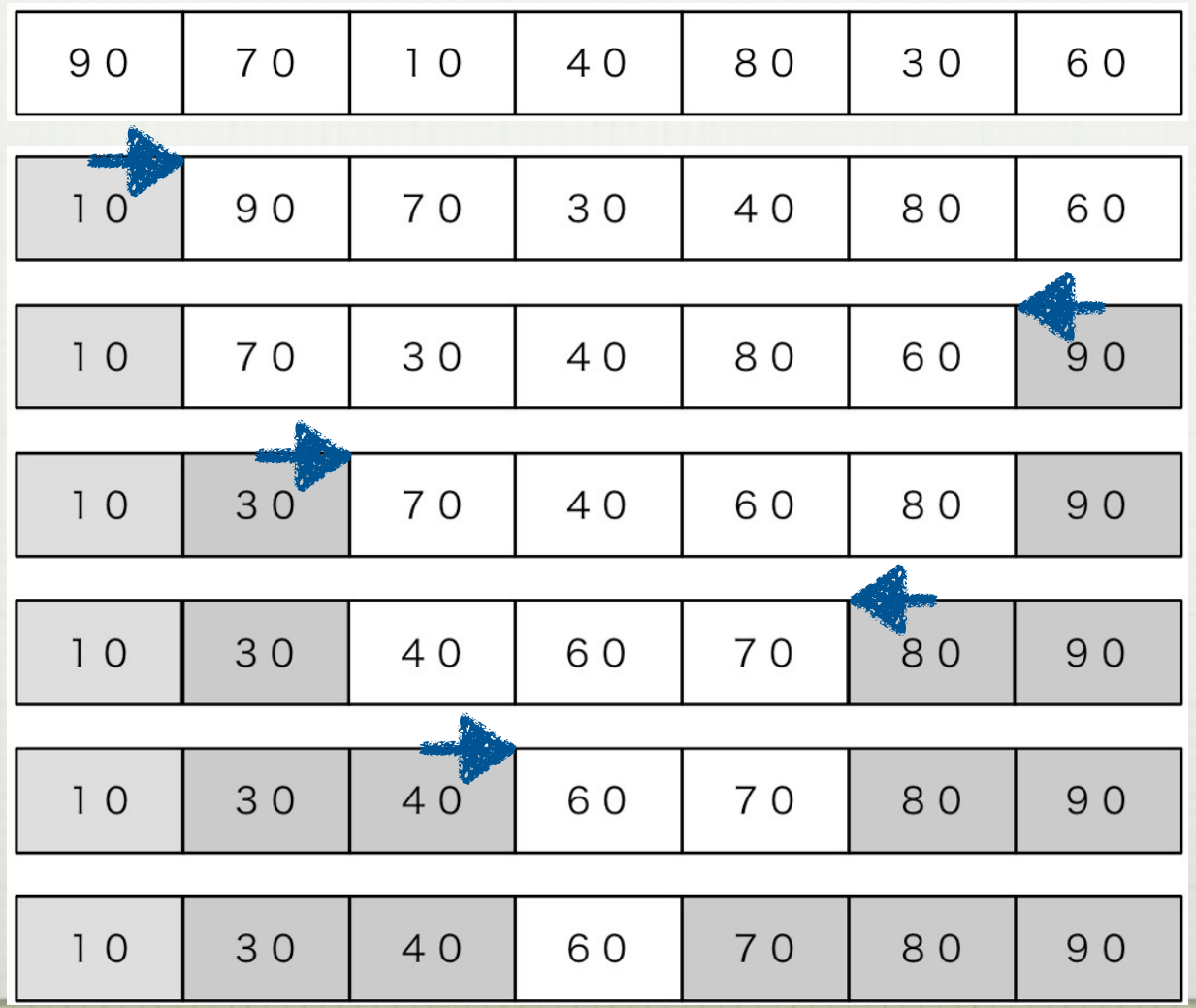


練習問題12-4

比較回数は変わらないが，
交換回数が少なくなる．

以下のデータ系列を双方向バブルソートで昇順に整列しなさい．

答え



比較：6回

比較：5回

比較：4回

比較：3回

比較：2回

比較：1回

課題12-2

- 課題12-1で用いたデータ集合に対して，双方向バブルソートを行い，英語・数学・物理の合計点順（降順）に整列して，整列した順に氏名および各科目の点数と合計点を表示するプログラムを作成しなさい．

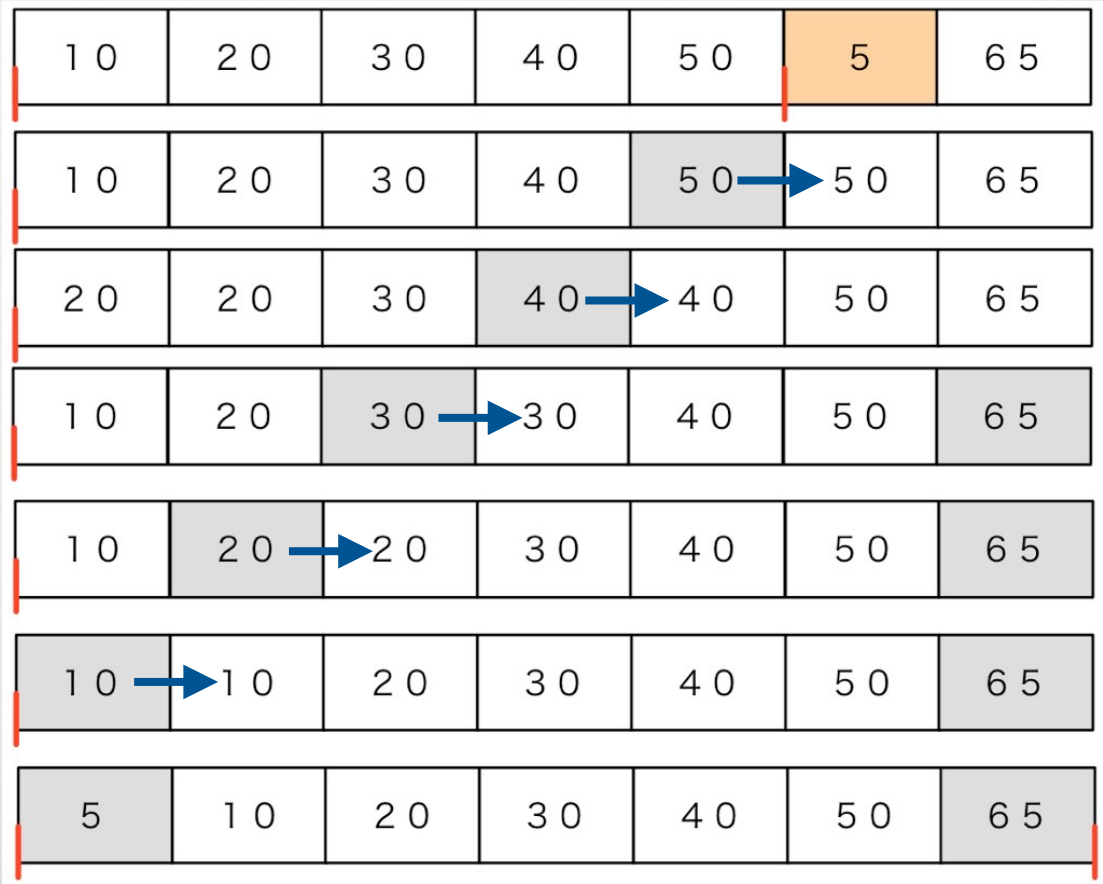
ソート法の分類

	ソート法	計算量
基本形	基本**交換**法（バブル・ソート）	$O(n^2)$
	基本**選択**法（直接選択法）	
	基本**挿入**法	
改良型	クイック・ソート（改良交換法）	$O(n\log_2 n)$
	ヒープ・ソート（改良選択法）	
	シェル・ソート（改良挿入法）	$O(n^{1.2})$

改良挿入法：シェルソート

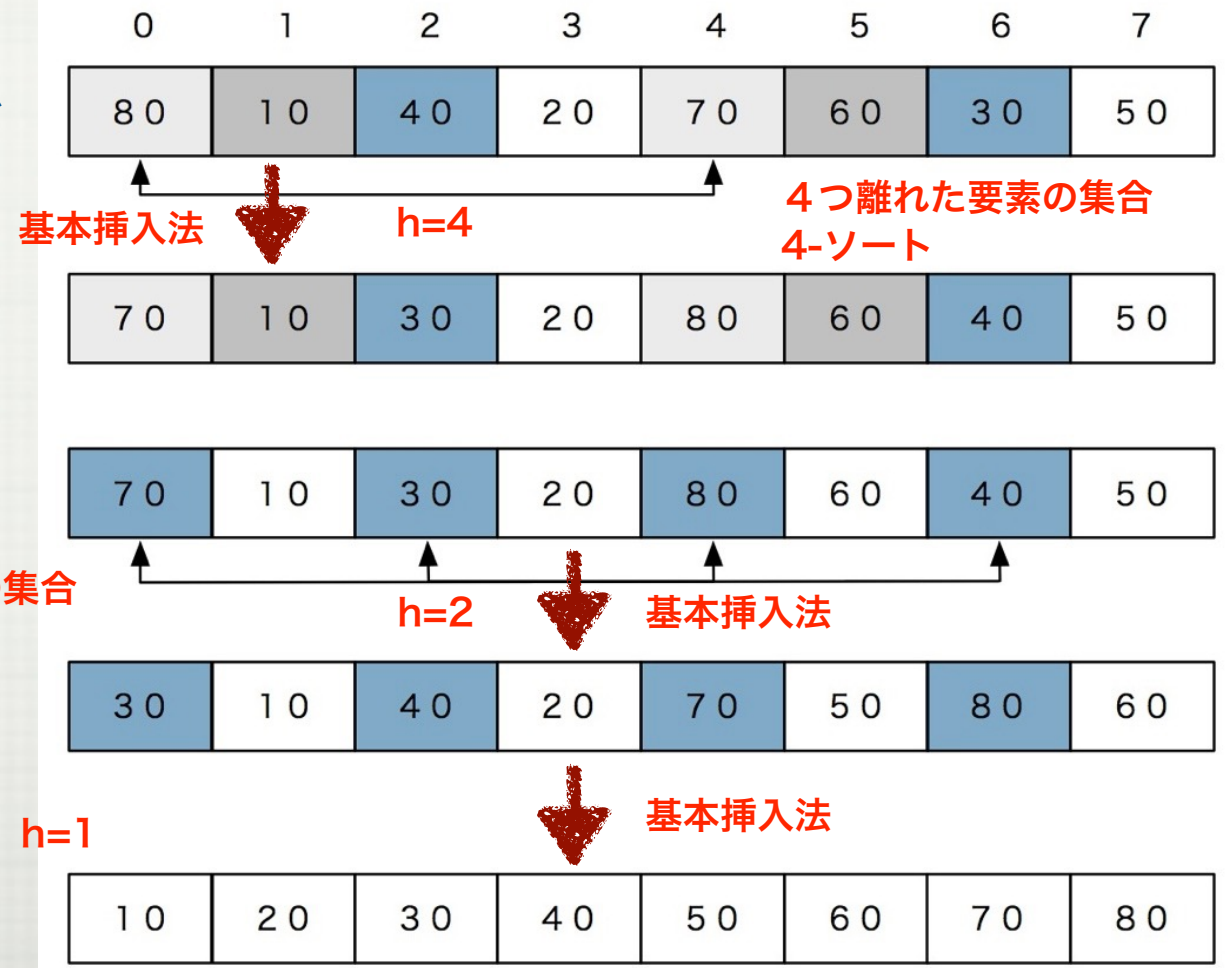
基本挿入法では...

- 整列に近い状態では高速
- 挿入先が離れている場合
  交換回数が増える



改良挿入法：シェルソート

- 距離h離れた要素の集合で基本挿入法による整列
- 距離を縮めて繰り返す
- h=1：基本挿入法と同じ



シェルソートでのhの設定

- h=..., 121, 40, 13, 4, 1
  h= h*3+1
  h < n/9 (n: 配列の要素数)
- 通常は，n/9を超えないhの最大値を求め，1/3ずつ減らす

課題12-3

- 要素数50の配列データとして，0～99までの乱数を入力し，
  シェルソートによって整列するプログラムを作りなさい(ex12-3.c).
  なお，hの値ごとに整列の途中結果を表示するとともに，最終的な比較・交換回数を表示するようにしなさい.

  ヒント．乱数を発生する関数: rand ()

レポート提出

- 課題12-1～12-3
- 提出期限：7月15日　0:00（7月14日まで）
- 提出先：Webclass