プログラミング演習Ⅰ
課題3（第2回）

宇都宮大学工学部情報工学科

概要

- 第1回　単純な整列アルゴリズム（1）
  - C言語の復習（配列，マクロ定義，代入演算子）
  - バブルソート
- 第2回　単純な整列アルゴリズム（2）
  - C言語の復習（3項演算子）
  - 選択ソートと挿入ソート
- 第3回　高速な整列アルゴリズム
  - CSVファイルの利用
  - クイックソート
  - 課題

C言語の復習

3項演算子(条件演算子)

条件式　？　式1　：　式2

```
if(a>0)
    b=1;
else
    b=0;
```

=

```
b = a>0 ? 1 : 0;
```

```
(a>0) ? (b=1) : (b=0);
```

多重条件演算子

```
if(a>1000)
    b=1000;
else if(a>100)
    b=100;
else if(a>10)
    b=10;
else
    b=0;
```

=

```
b = (a>1000) ? 1000 :
    (a>100) ? 100:
    (a>10) ? 10: 0;
```

3項演算子の練習

- (age>=20) ? printf(“大人¥n”) : printf(“子供¥n”);
- #define MIN(a,b) (((a)<(b))?(a):(b))
  - z = MIN(x,y);
- #define ABS(x) ((x>0)?(x):(-x))
  - z = ABS(x);

練習問題(1)

- #define MAX(a,b) (((a)>(b))?(a):(b))を用いて配列a[]の要素のうち最大の要素を探す関数int find_max_value(const int a[], int n)を実装せよ。

練習問題(2)

- 引数がアルファベットであり、大文字なら小文字に、小文字なら大文字に変換し、アルファベットでなければそのまま返す関数int lu_exchange(int c)を条件演算子を用いて実装せよ（string.hとctype.hは利用可）
  - ヒント
    - int islower(int c); 文字cが小文字(a～z)なら真を返す
    - int isupper(int c); 文字cが大文字(A～Z)なら真を返す
    - int tolower(int c); 文字cが大文字なら小文字に変換して返す
    - int toupper(int c); 文字cが小文字なら大文字に変換して返す

選択ソート

選択ソートのアルゴリズム

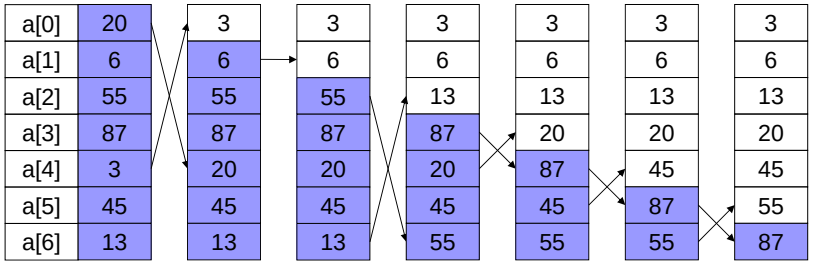
- 整列されていない部分から最小の要素を選び出し、それをその部分の先頭に持っていく

1. a[0]からa[n-1]のうち最小の要素を探し、それをa[0]と交換
2. a[1]からa[n-1]のうち最小の要素を探し、それをa[1]と交換
3. a[2]からa[n-1]のうち最小の要素を探し、それをa[2]と交換
4. a[3]からa[n-1]のうち最小の要素を探し、それをa[3]と交換

n-1. a[n-2]からa[n-1]のうち最小の要素を探し、それをa[n-2]と交換

選択ソートの過程

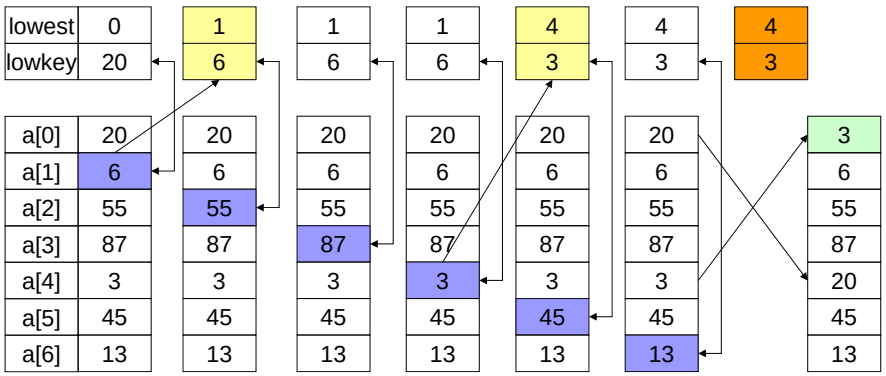
整列されていない部分



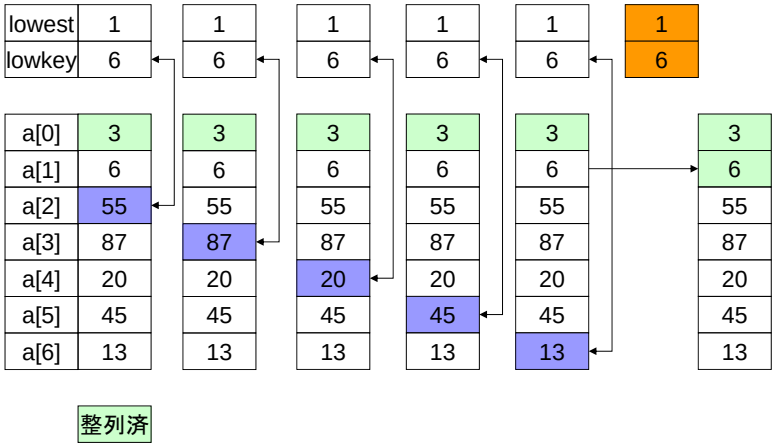
どのようにして最小の要素を探し出すか？

- 基準となる要素(lowkey)を保持しておき，それと各要素(a[i])を比較して，a[i]がlowkeyより小さければ，lowkeyをそのa[i]と置き換える．
- 置き換えが起こったとき，その要素番号(i)も保持（lowest）しておく必要がある．

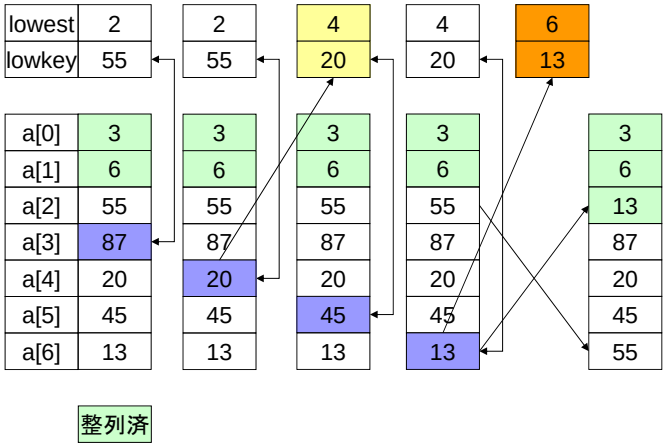
配列の中で最小の要素を得る方法（1回目）



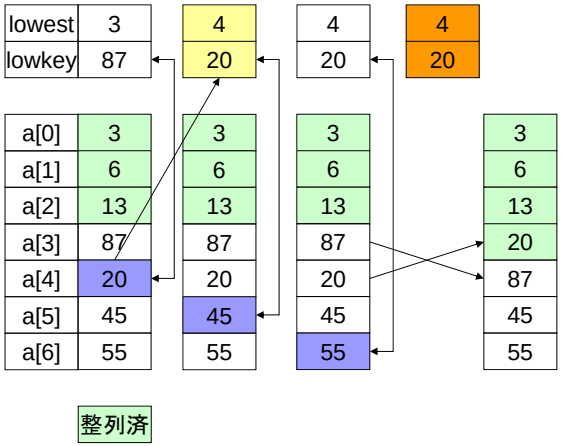
配列の中で最小の要素を得る方法（2回目）



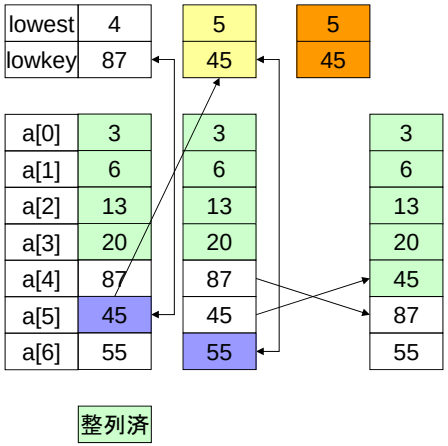
配列の中で最小の要素を得る方法（3回目）



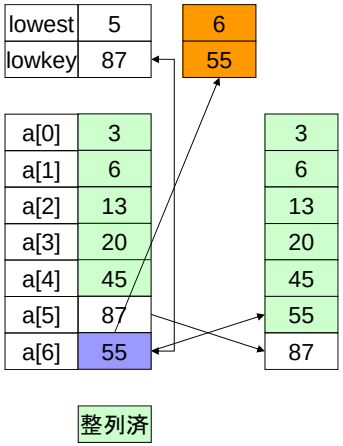
配列の中で最小の要素を得る方法（4回目）



配列の中で最小の要素を得る方法（5回目）



配列の中で最小の要素を得る方法（6回目）



- 変数lowestは必要か?
  整列されていない部分から，最小の要素が見つかったとき，それをその部分の先頭に持っていく（交換）ために必要

練習問題(3)

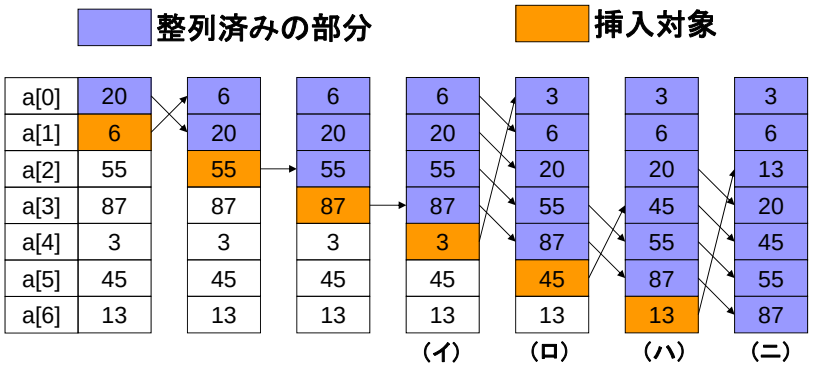
- 選択ソートを行う関数void selection_sort(int a[], int n)を実装せよ

挿入ソート

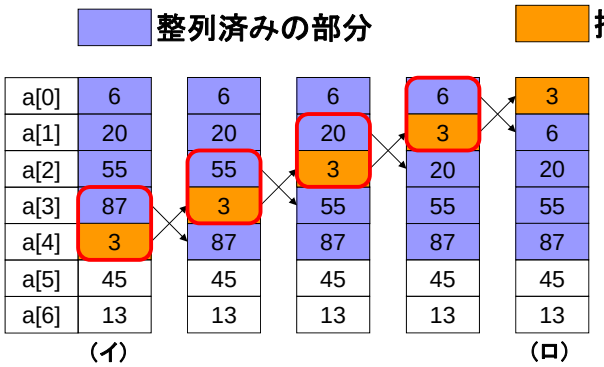
挿入ソートのアルゴリズム

- 配列のうち一部分を整列済みの状態にしておき、残りの要素を1つずつその中の適切な位置に挿入する。
  - トランプの手札を並び替えるに手順と同様

挿入ソートの手順

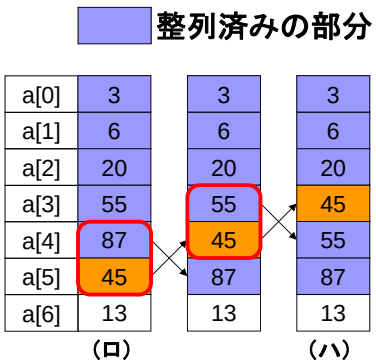


挿入ソートの手順(イ)⇒(ロ)



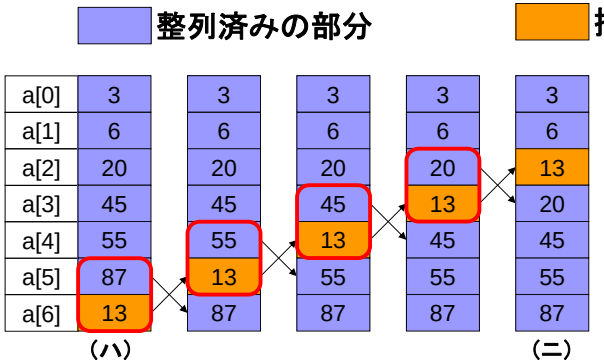
適切な位置が見つかったら終了

挿入ソートの手順(ロ)⇒(ハ)



適切な位置が見つかったら終了

挿入ソートの手順(ハ)⇒(ニ)

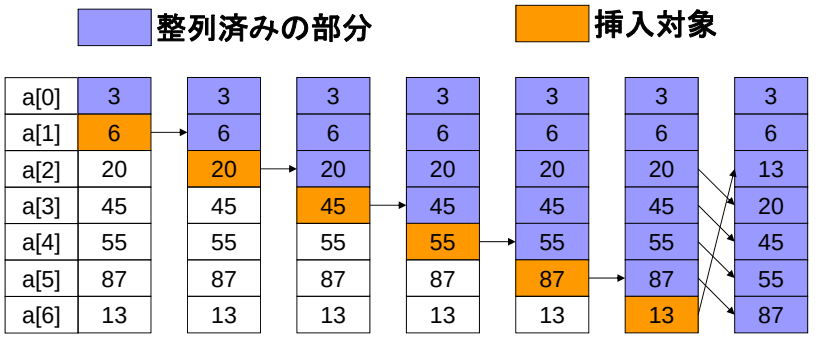


適切な位置が見つかったら終了

挿入ソートの特徴

- ほとんどの要素がすでに整列済みの配列に対して適している.
- 整列済みの配列に対して，新たに要素を付け加えたものを整列する場合に適している.

整列済みの配列に新たな要素を追加した配列の挿入ソートの手順



練習問題(4)

- 挿入ソートを行う関数void insertion_sort(int a[], int n)を実装せよ