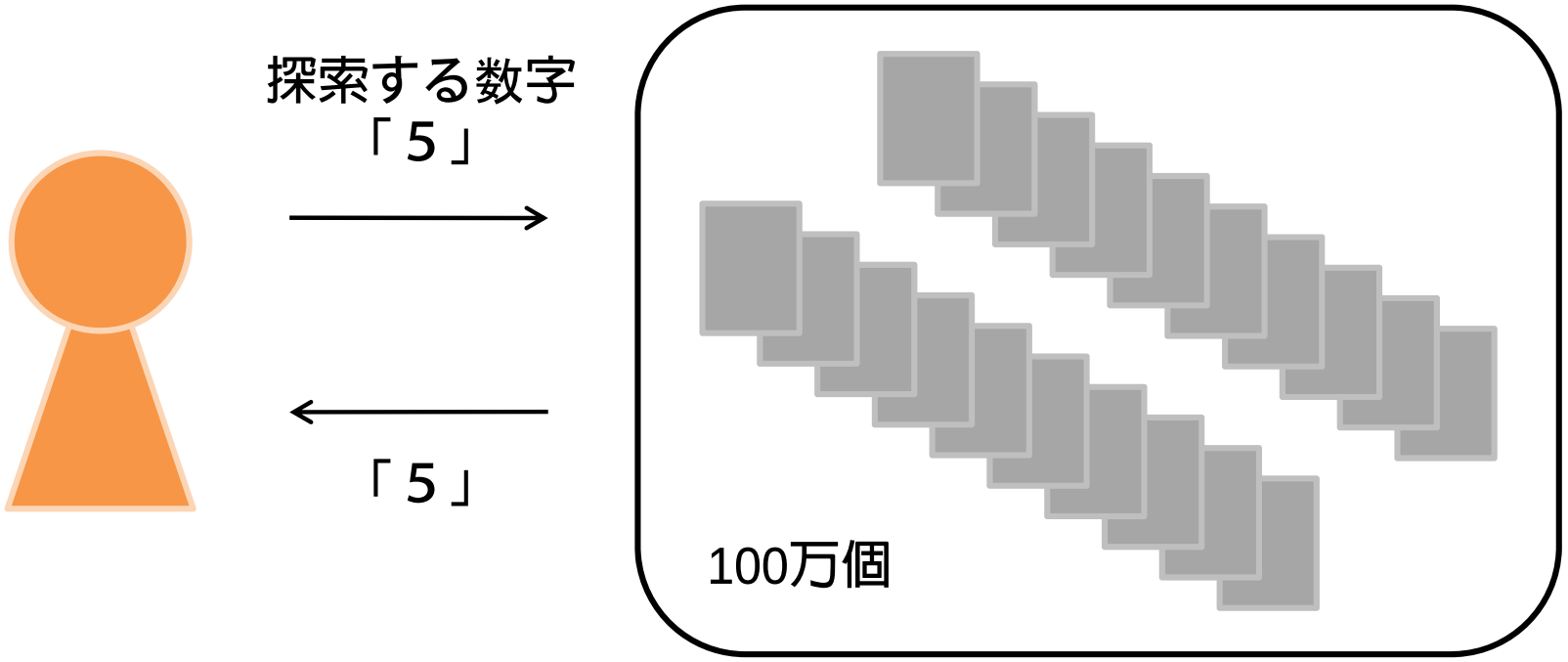
データ構造入門及び演習
10回目:二分探索

2014/06/20

担当:見越　大樹
61号館304号室

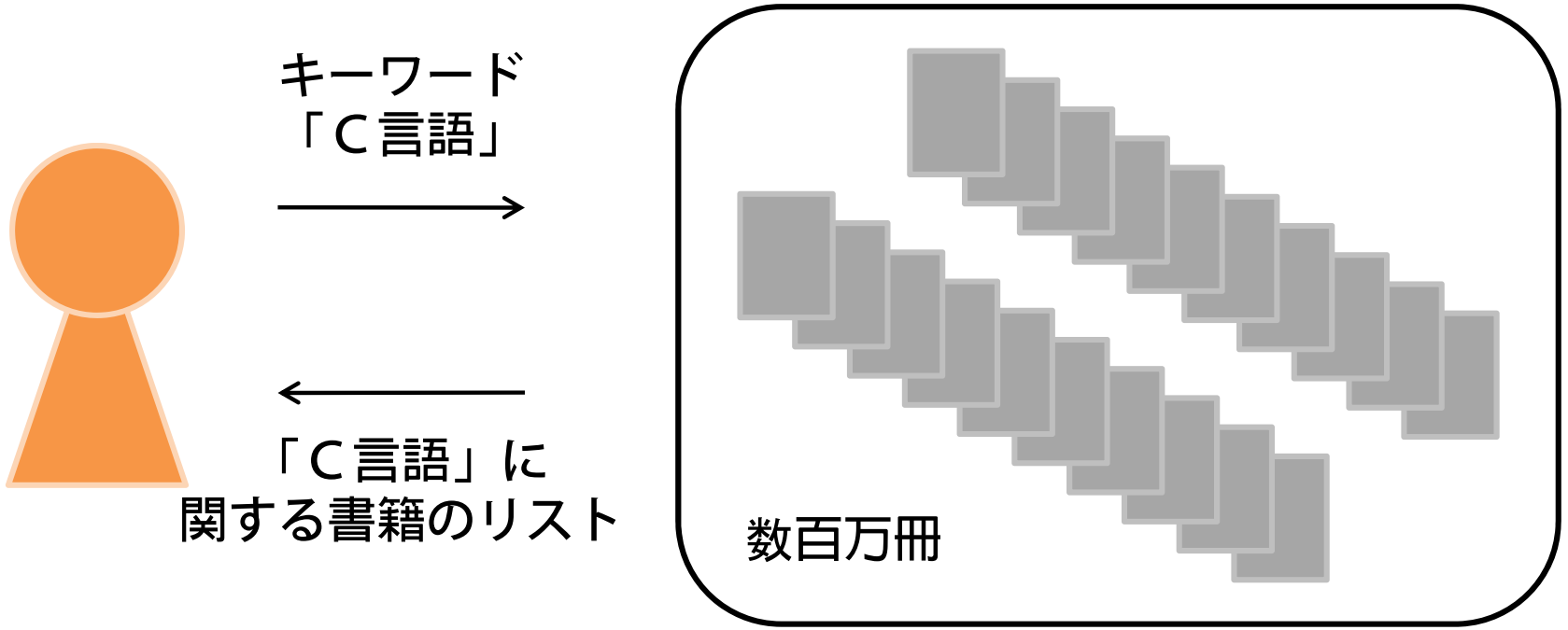
データの探索

- 複数のデータから，見つけたいデータを探すこと
  - 例） 配列の中から数字「５」をさがす



データの探索

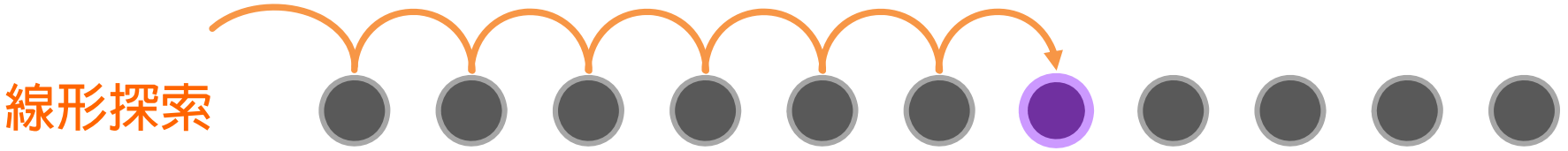
- 例：オンライン書店での書籍の探索



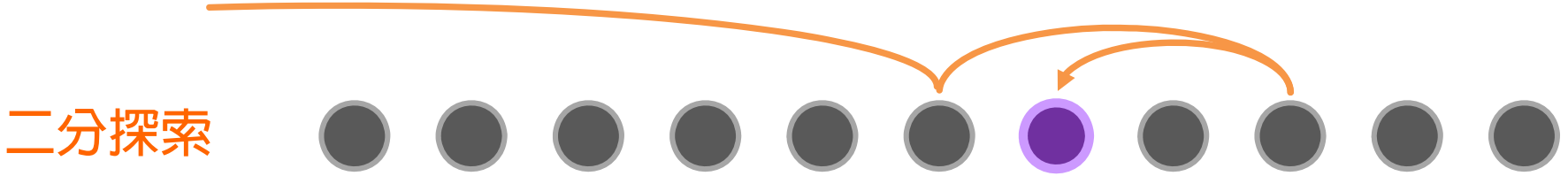
- データ量が多いほど高速なアルゴリズムが必要
  - データベースの検索，WEB検索，WORD検索機能

探索 (Search)

- よく知られた探索アルゴリズム:
  - 線形探索 (Linear Search)　1年次に学習済み
    - for文を使って先頭から順番に探査する
    - 探索の基本，遅い，ソート済みでないデータにも適用できる
  - 二分探索 (Binary Search)　本日学習
    - 高速，ソート済みデータにのみ適用できる



前から順に一つ一つ探していく



対象を半分ずつに絞って探していく

線形探索

- 配列要素の先頭から順に探していく方法

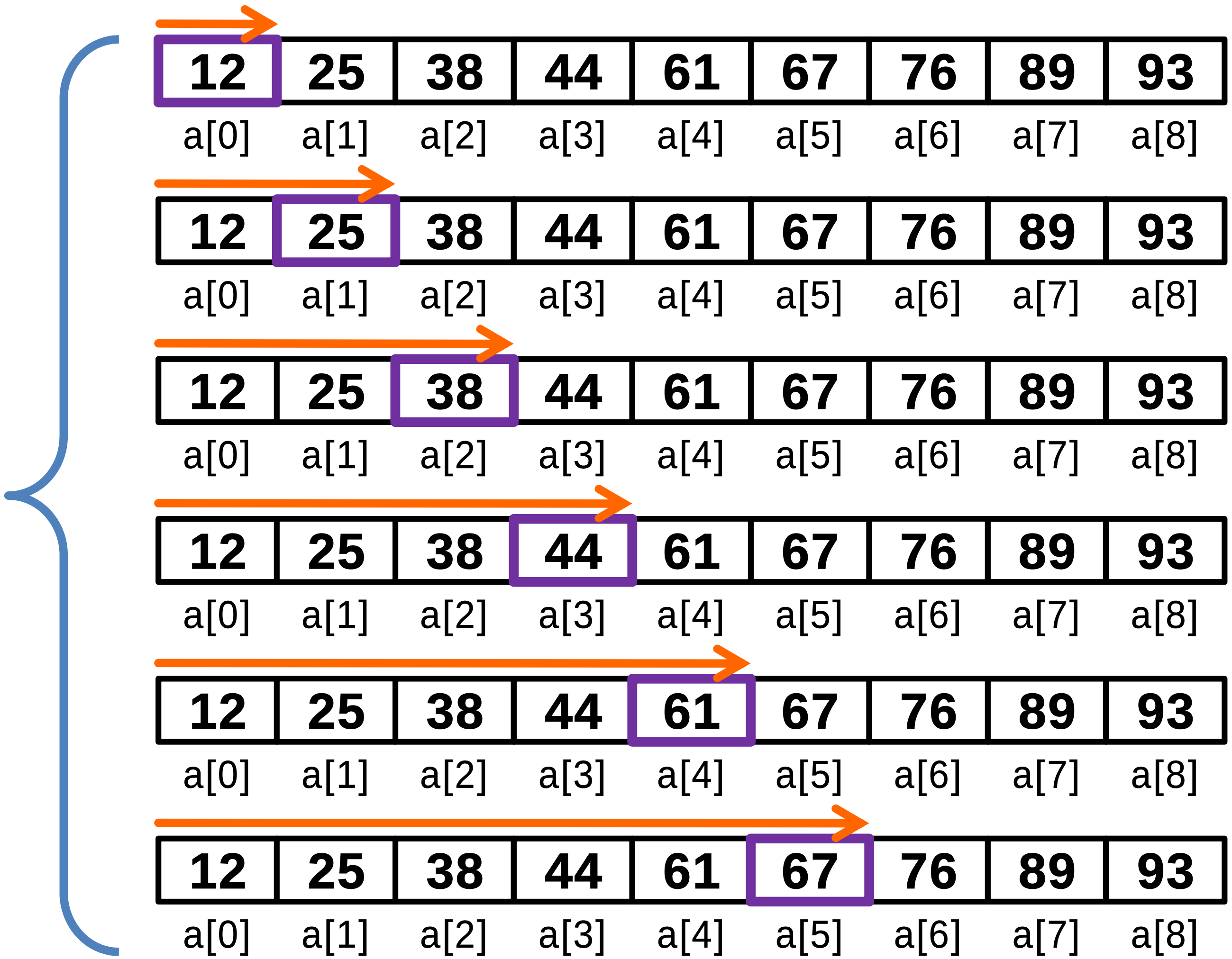
探索対象データ：

12	25	38	44	61	67	76	89	93

67

見つけたい値

先頭から一つずつ
順に比較していく



線形探索プログラム

```
#include  <stdio.h>
#define  NUM 9
int  linear_search( int a[], int key );

void  main()
{
    int  i,key,id;

    //  入力データ
    int  a[] = { 12, 25, 38, 44, 61, 67, 76, 89, 93 };

    //  見つけたい数値を「key」に入力
    printf("見つけたい数値を入力してください:");
    scanf( "%d", &key );

    //  線形探索
    id = linear_search( a, key );

    //  結果の表示
    if( id == -1 )
        printf( "探索に失敗しました. ¥n¥n" );
    else
        printf( "%dはa[%d]に見つかりました. ¥n¥n", key, id );
}
```

```
int  linear_search( int a[], int key )
{
    int  i;

    //  見つかったら配列番号を返す
    for( i=0;i<NUM;i++ ){
        if( a[i] == key ){
            return  i;
        }
    }
    //  見つからなかったら-1を返す
    return  -1;
}
```

線形探索の欠点

探査対象データ:

1から100までの整数列から，40を探索する．ただし，整数列は昇順（小さい順）に並んでいるものとする．

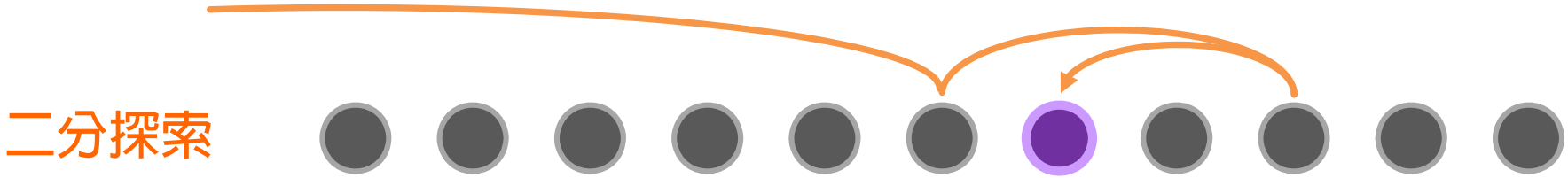
欠点:

線形探索の場合，1から順に40と比較し，40番目に40が見つかる．探索が完了するまで比較が40回必要．計算が遅い！

データが並んでいる場合，線形探索は効率が悪い！

探索 (Search)

- 二分探索 (Binary Search)
  - 高速，ソート済みデータにのみ適用できる



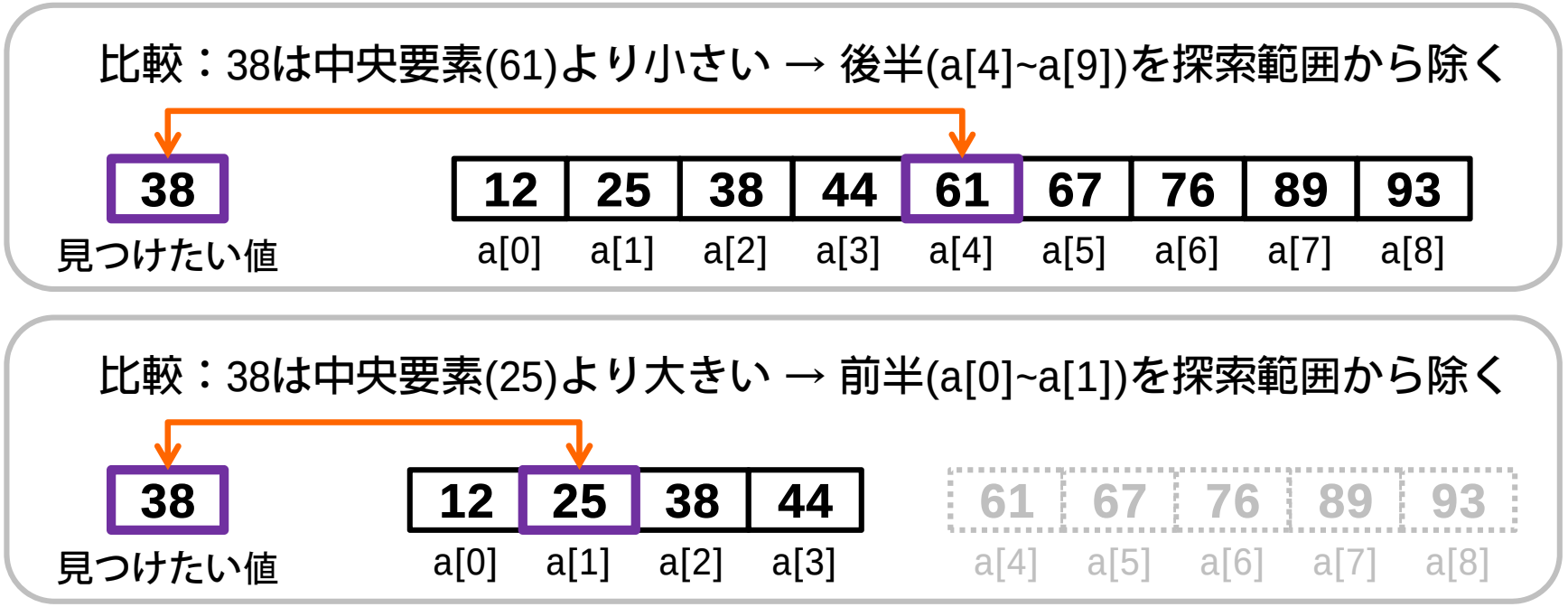
対象を半分ずつに絞って探していく

二分探索

- 整列済データを2つに分け，目的の探索範囲を絞り込む方法

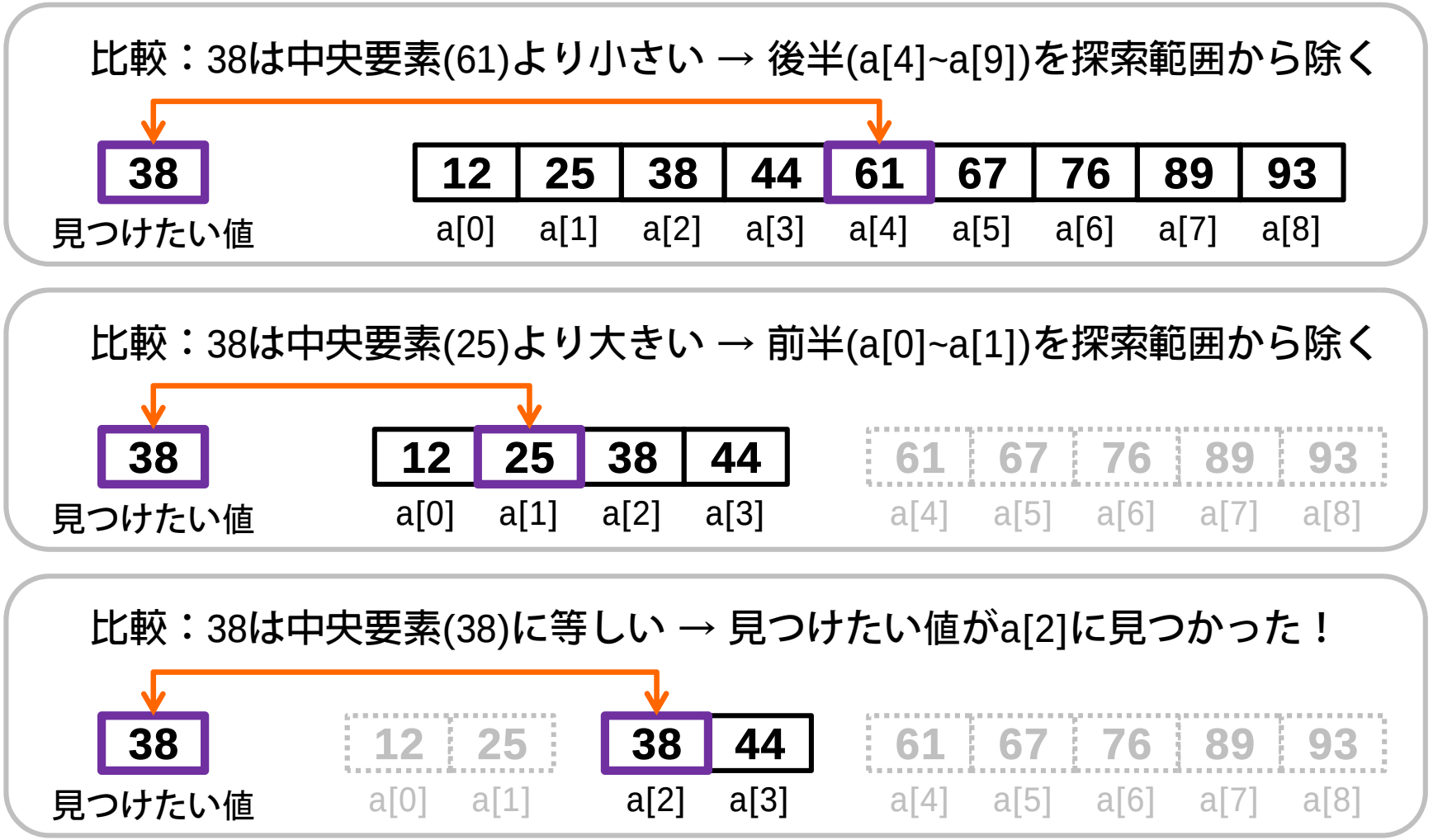
二分探索

- 整列済データを2つに分け，目的の探索範囲を絞り込む方法



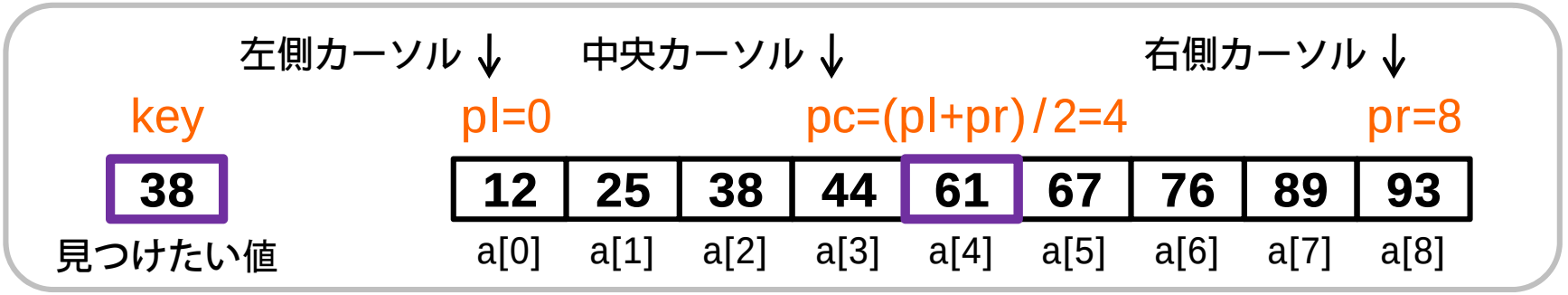
二分探索

- 整列済データを2つに分け，目的の探索範囲を絞り込む方法



プログラム上で探索範囲を絞り込む方法

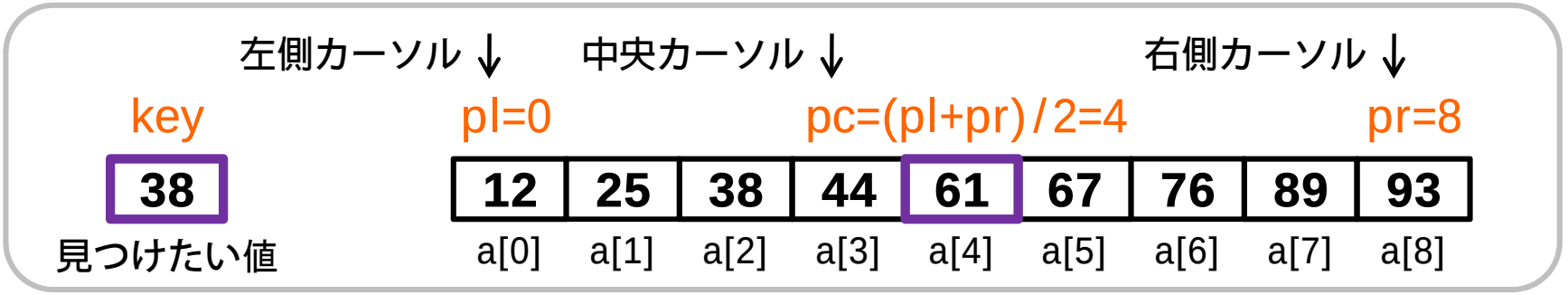
- 探索対象配列の最初と最後の添え字を確保しておく！

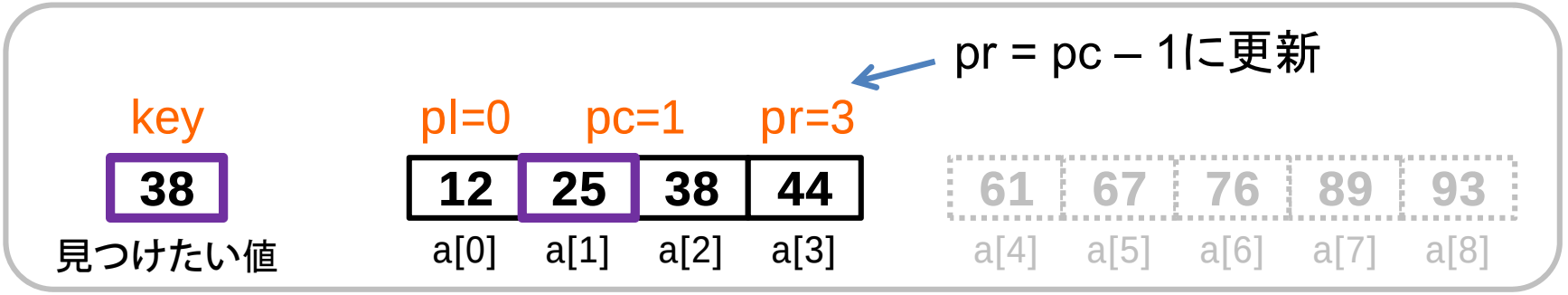


最初(pl)と最後(pr)を格納しておけば，中央(pc)は自動的に決まる
見つけたい値(key)は別に格納しておく

プログラム上で探索範囲を絞り込む方法

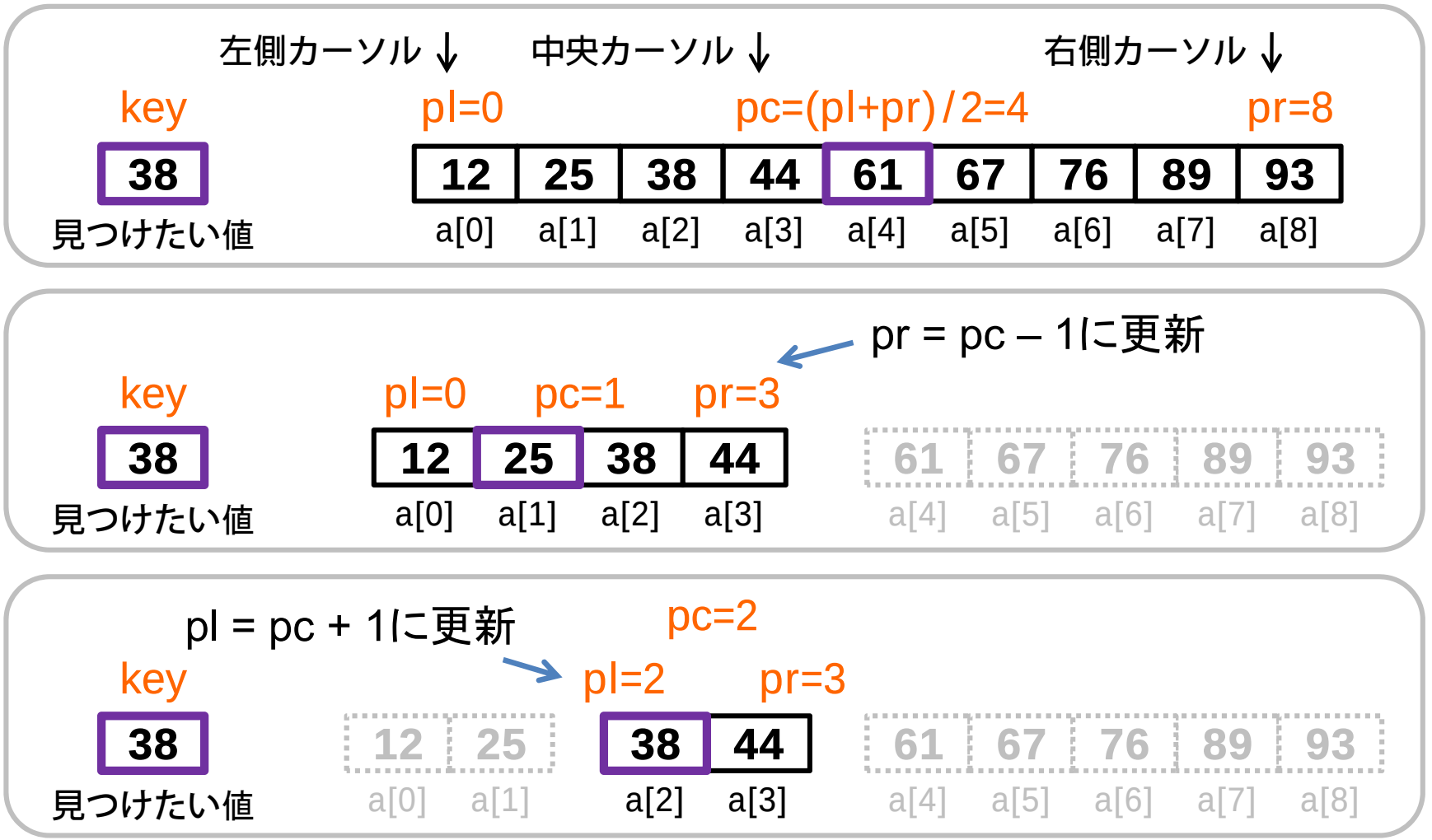
- 探索対象配列の最初と最後の添え字を確保しておく！





プログラム上で探索範囲を絞り込む方法

- 探索対象配列の最初と最後の添え字を確保しておく！

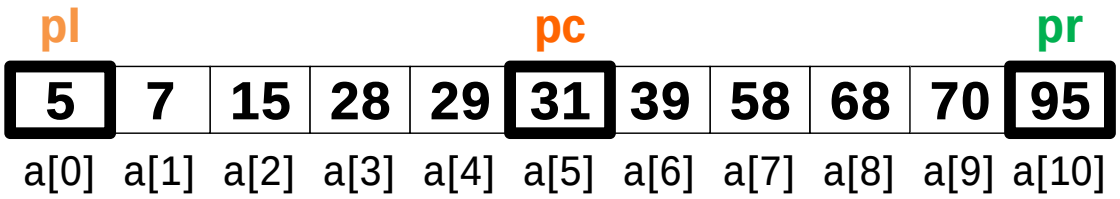


二分探索プログラム（概要）

- 初期化
  - 配列の最初と最後の番号を格納　pl=0, pr=NUM-1;
- 繰り返し処理
  1. 配列中央の番号を格納　pc=(pl+pr)/2;
  2. 中央が見つけたい値よりも小さければ，探索範囲縮小 (小さい方を削除)
     if(a[pc] < key)　pl=pc+1;
  3. 中央が見つけたい値よりも大きければ，探索範囲縮小 (大きい方を削除)
     else if(a[pc] > key)　pr=pc-1;
- 終了条件
  - 条件1（探索失敗）: pl > pr
  - 条件2（探索成功）: a[pc] == key;

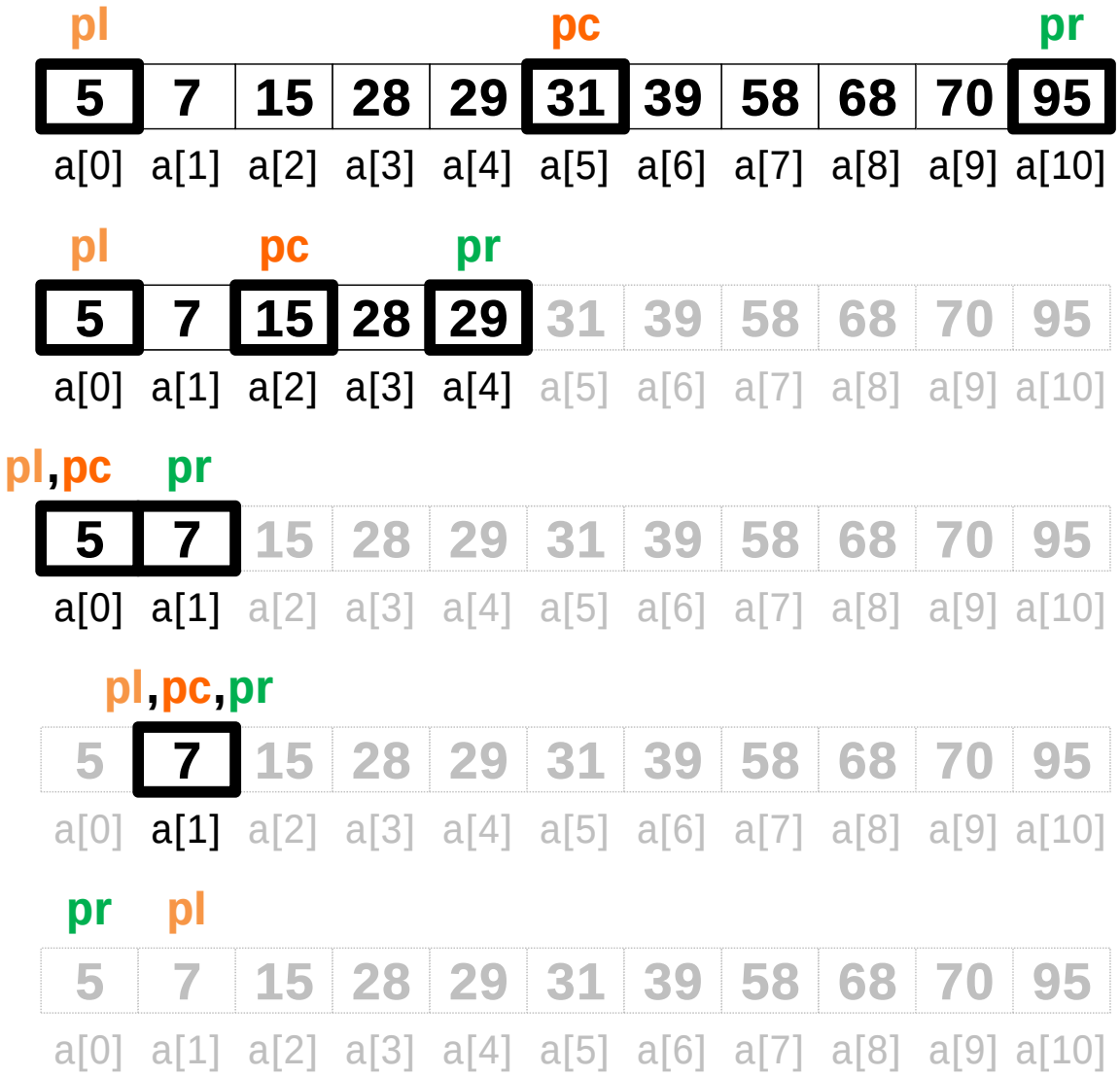
二分探索アルゴリズム（成功例：key=39）

1. pl=0, pr=10, pc=5;
2. a[pc]<key
3. pl=6, pr=10, pc=8;
4. a[pc]>key
5. pl=6, pr =7, pc=6;
6. a[pc]==key
   (→ 成功)



二分探索アルゴリズム（失敗例：key=6）

1. pl=0, pr=10, pc=5;
2. a[pc]>key
3. pl=0, pr=4, pc=2;
4. a[pc]>key
5. pl=0, pr =1, pc=0;
6. a[pc]<key
7. pl=1, pr=1, pc=1;
8. a[pc]>key
9. pl=1, pr=0;
   (pl>pr → 失敗)



二分探索プログラム

```
#include  <stdio.h>
#define  NUM 9
int  binary_search( int  a[], int  key );

void  main()
{
    int  i, key, id;

    //  入力データ
    int  a[] = { 12, 25, 38, 44, 61, 67, 76, 89, 93 };

    //  見つけたい数値を「key」に入力
    printf("見つけたい数値を入力してください:");
    scanf( "%d", &key );

    //  二分探索
    id = binary_search( a, key );

    //  結果の表示
    if( id == -1 )
        printf( "探索に失敗しました. ¥n¥n" );
    else
        printf( "%dは%dに見つかりました. ¥n¥n", key, id );
}
```

```
int  binary_search( int  a[], int  key ) {
    int  pl, pr, pc;

    //  初期化：配列の最初と最後の番号を格納
    pl = 0;    pr = NUM-1;

    //  繰り返し処理
    while(1){

        //  終了条件1(左右のカーソルが逆転，探索失敗)
        if( pl > pr )  return -1;

        //  1. 配列中央の番号を格納
        pc = (pl+pr)/2;

        //  2. 中央が見つけたい値よりも小さければ
        if( a[pc] < key )
            pl = pc+1;    //  探索範囲縮小(小さい方を削除)

        //  3. 中央が見つけたい値よりも大きければ
        else if( a[pc] > key )
            pr = pc-1;    //  探索範囲縮小(大きい方を削除)

        //  終了条件2(中央とkeyが一致，探索成功)
        else if( a[pc] == key )    //
            return  pc;    //  配列中央の番号を返す
    }
}
```

二分探索の利点

探査対象データ:

1から100までの整数列から，40を探索する．ただし，整数列は昇順（小さい順）に並んでいるものとする．

利点:

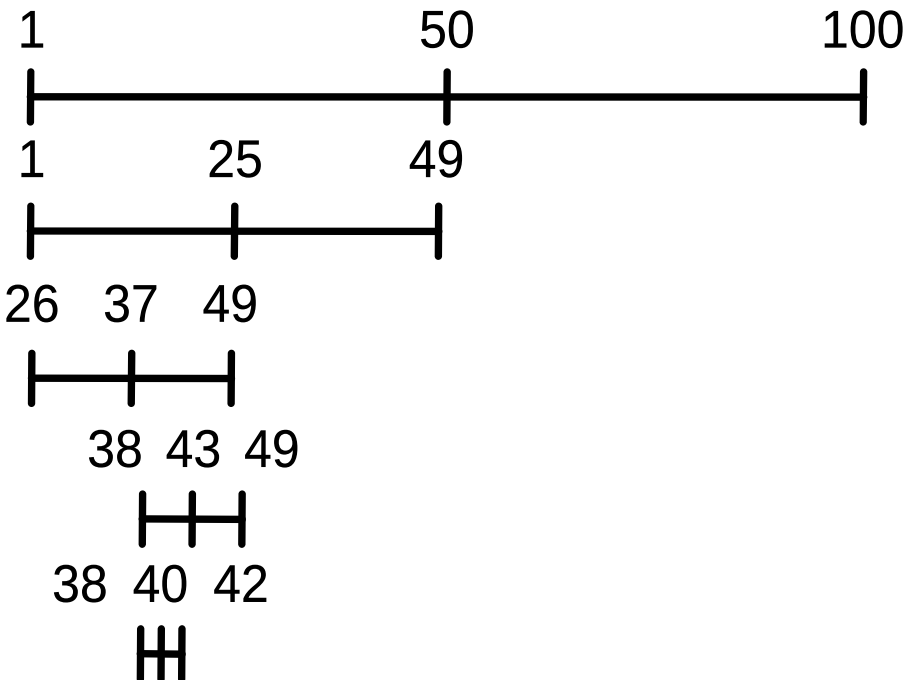
5回の比較で40が探索可能．計算が速い！

(線形探索では40回の比較)

毎回探索範囲を半分に限定

平均的に$\log_2 N$回の比較で探索

(Nはデータの個数)



二分探索のまとめ

- 線形探索に比較すると非常に高速な探索が可能
- ソート済みデータのみ適用可能であり，事前にデータの整列処理が必要
- 探索範囲の絞り込み手順を理解
  - 右カーソル「pr」，左カーソル「pl」の動き
  - 中央カーソル「pc」の設定