アルゴリズム論 4,5,6

探索

- 線形探索
- ２分木探索
- ハッシュ探索

2分探索の考え方1

問題：

1から100までの整数から40を探索する。

ただし整数列は昇順に並んでいるものとする

線形探索の場合：

1から順に40と比較していき、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

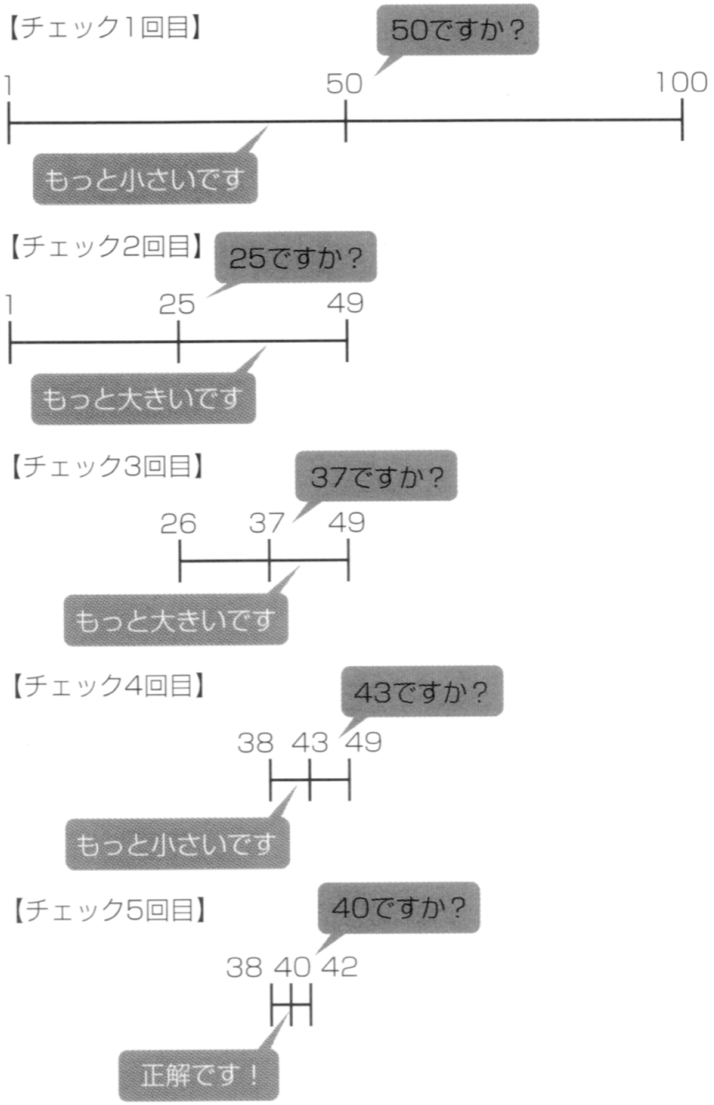
17

2分探索の考え方2

2分探索の場合：

ある数より大きいか小さいかをチェックする。

5回の比較で40が探索可能

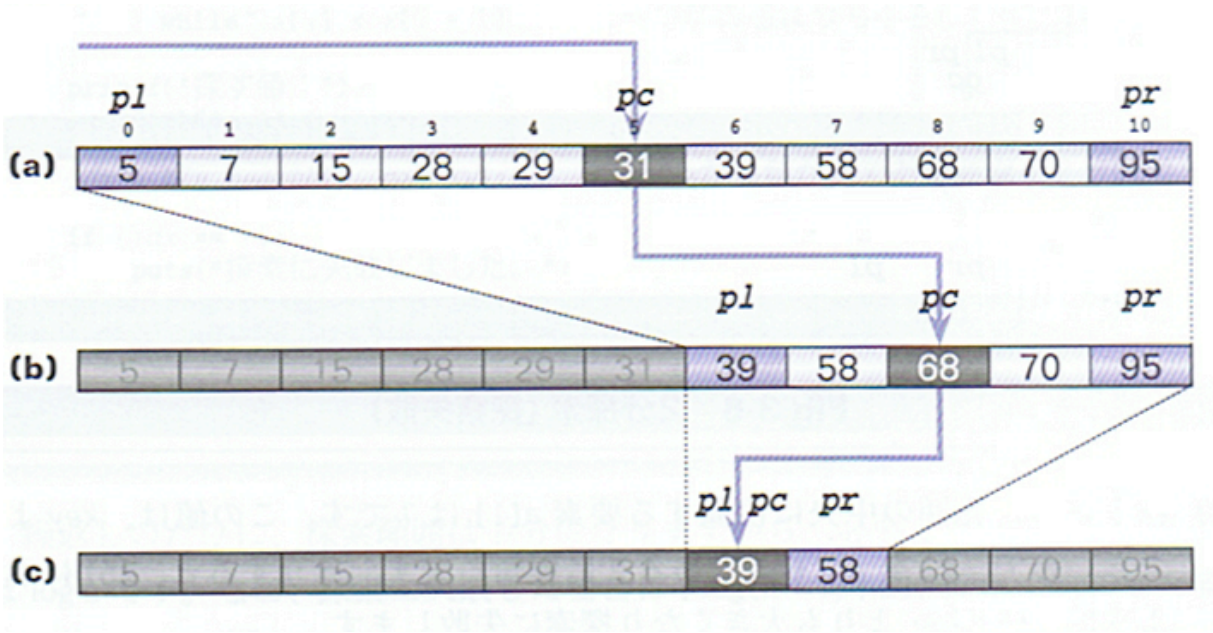


18

2分探索のアルゴリズム1

前提

- a[]:昇順整列している要素数nの配列
- ある数(key)を探索する
- 探索範囲の先頭a[pl]　末尾a[pr]、中央a[pc]



19

2分探索のアルゴリズム2

探索開始時

pl=0, pr=n-1, pc=(n-1)/2

(1) **a[pc]=key ならば探索終了**

(2) **a[pc]<key ならばpl=pc+1, pc=(pl+pr)/2**

(3) **a[pc]>key ならばpr=pc-1, pc=(pl+pr)/2**

探索範囲縮小

探索範囲縮小

(2)または(3)を繰り返す

探索終了条件

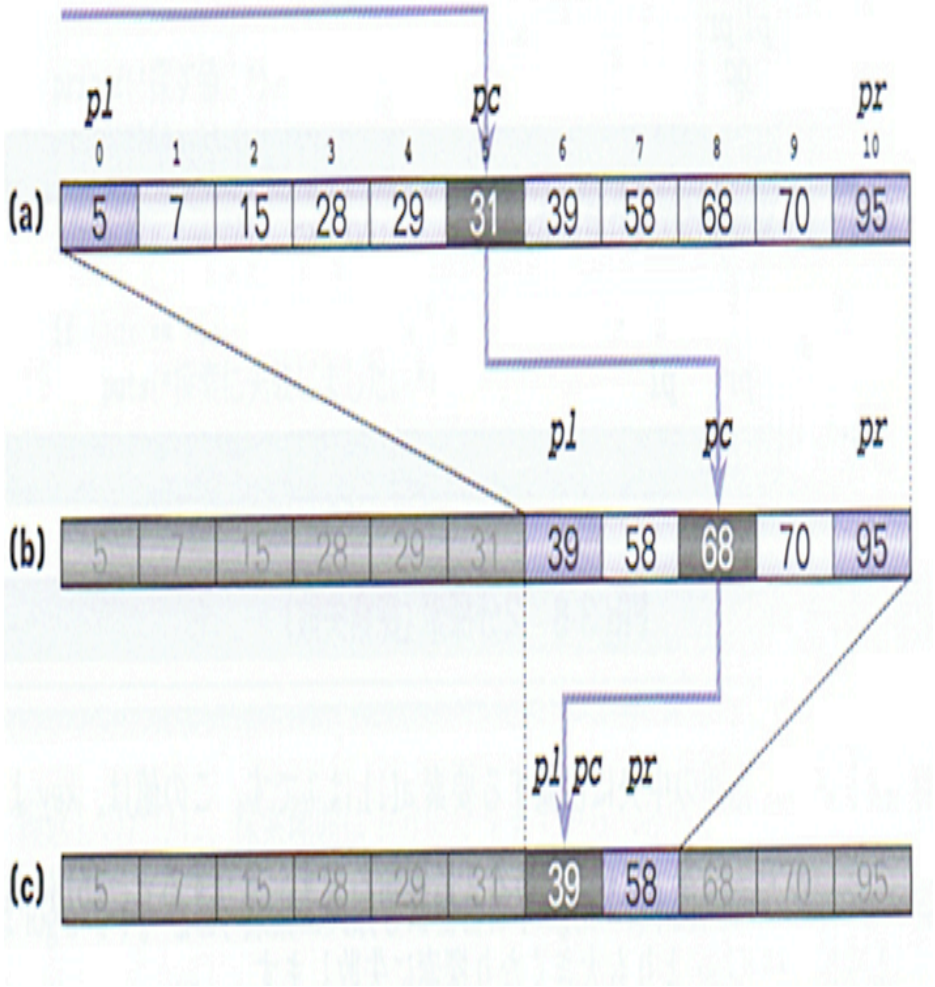
条件1(探索成功)：a[pc]=key

条件2(探索失敗)：pl>pr

20

2分探索のアルゴリズム3

探索　成功例



39を探索(key=39)

(1)pl=0,pr=10,pc=5

(2)a[pc]<key

(3)pl=6,pr=10,pc=8

(4)a[pc]>key

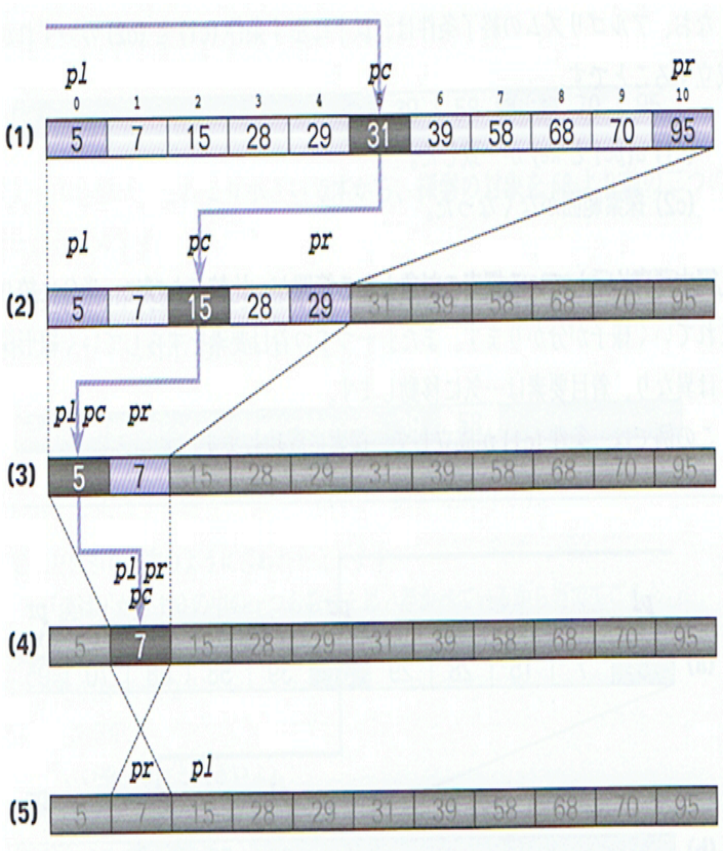
(5)pl=6,pr=7,pc=6

(6)a[pc]=key　→探索成功

21

2分探索のアルゴリズム4

探索 失敗例



6を探索(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)** →探索失敗

22

2分探索のプログラム(メイン)

```
#include <stdio.h>
int bin_search(int a[], int n, int key); /* 関数プロトタイプ */
#define NUM 7

int main(void)
{
   int         i,ky,idx;
   int         x[NUM];

   printf(" Input integer number %d times ¥n",NUM); /* データ入力 */
   for (i=0;i<NUM;i++)      {
       printf("x[%d]:",i);
       scanf("%d",&x[i]);
   }
   printf("Number to search:");  /* 探索数値入力  */
   scanf("%d",&ky);

   idx=bin_search(x,NUM,ky);       /* 2分探索  */
   if (idx==-1)
       printf("Searching was failed!¥n");  /* 探索失敗  */
   else
       printf("%d is located at %d ¥n",ky,idx); /* 探索成功  */
   return(0);
}
```


23

2分探索のプログラム（関数）

```
int bin_search(int a[], int n, int key)
{
   int pl=0;
   int pr=n-1;
   int pc;

   do  {
       pc=(pl+pr)/2;
       if (a[pc]==key)
              return(pc);
       else if (a[pc]<key)
              pl=pc+1;
       else
              pr=pc-1;
   } while(pl<=pr);
   return (-1);
}
```

→初期化

→探索成功

→探索範囲縮小
(大きい方に)

→探索範囲縮小
(小さい方に)

→探索失敗

24

2分探索のプログラム 実行結果

```
Input integer number 7 times
x[0]:15
x[1]:27
x[2]:39
x[3]:77
x[4]:92
x[5]:108
x[6]:121
Number to search:39
39 is located at 2
```


25

演習問題(講義時間内で実施)

- 2分探索を行うプログラムのソースコードを入力し実行形式ファイルを作成する
  - メイン（線形探索のメインを流用）
  - 2分探索関数
- データを入力し、実行結果を確認する
- 計算量を検討するためにカウンタをつける

26

計算量を算出する機能を追加

```
int bin_search(int a[], int n, int key)
{




}
```


27

2分探索の特徴

- 探索の回数毎に探索範囲が**半分**に減少
- 探索可能な数に注目
  - １回の探索：１　個
  - ２回の探索：３　個
  - ３回の探索：７　個
  - ｊ回の探索：2^j-1　個

28

2分探索の計算量

k回のステップで探索可能なデータ数

$$n=2^k-1 \Rightarrow k=\log_2[n+1]$$

オーダ　$O(\log_2 n)$

非常に高速である

29

課題問題2(レポート提出要)

課題の目的:線形探索と2分探索の計算量の違いを理解する

- 線形探索の計算量を算出するプログラムを入力し実行する。
- 2分探索の計算量を算出するプログラムを入力し実行する。
- 入力データ数を変化させて計算量を確認する。
  - 入力データ数、データ入力方法等のプログラム修正は各自が工夫すること
- 線形探索と2分探索のオーダを確認および考察する。

レポートを作成し次週(5/27)に提出すること

レポートのフォーマットは自由とする

30