表探索

表探索とは

- 表の形で蓄えられている情報の中から条件に合った要素を取り出してくる操作である
  - 例：学生名簿から、学籍番号が100100のすべての情報（氏名、所属学科、成績など）を取り出してくる
- 表はよく配列で実現されるので、以降、「表」の代わりに直接「配列」の表現を用いる場合が多い
- 表探索をただ「探索」と呼ぶ場合が多い

用語

フィールド

学籍番号	氏名	所属課程	成績
000010	龍谷太朗	数理情報	100
000105	瀬田花子	電子情報	90
⋮	⋮	⋮	⋮
000100	大津三郎	知能情報	80

レコード

氏名を探索するなら氏名がキー（と呼ぶフィールド）で、
瀬田花子を探索するなら、瀬田花子がキーの値である

用語

- レコード：表中の個々のデータ
  - 例：学籍番号100100に関するデータ
- フィールド（項目）：レコードを構成する一つ一つの要素
  - 例：氏名、所属課程、成績など
- キー（と呼ぶフィールド）：探索の対象となるフィールド
  - 例：「所属課程」を対象に探索するなら、これがキー
- キーの値：「所属課程」がキーであれば、探索する具体的な課程、例えば「数理情報」がキーの値（キー値）

- 探索は、キーの値を指定して、それと等しい値のキーを持つレコードを選び出すという形で行われる。言い換えると、探索の入力はキーの値、出力はレコードである

探索の方法

- 線形探索
  - データを一つ一つ順に端から調べていくだけの素朴な探索法
- 2分探索
  - データをキーの値の大きさの順序に並ぶようにしておいて、その順序という特性を利用した比較的効率のよい探索法
- ハッシュ探索
  - キーの値を引数とし、ある関数を計算し、その関数値をデータの保存場所とし、探索もこの場所でデータを取り出せばよい最高速の探索法。計算量はO(1)である

線形探索

整数データを例とする

線形探索アルゴリズム

入力：配列a[0],...,a[n-1], キー値x
出力：キー値xがaの中の位置p（存在しなければ-1）
補助：i

？

計算量：O(n)

2分探索

整数データを例とする

考え方

- ソートされた配列からキー値を探す
  - 配列を2つに分け、キー値が配列の前半にあるか後半にあるかを調べる
  - 前半にあるなら、前半をさらに2つに分け、そのどちらにキー値があるかを調べる
  - 後半にある場合も同様である
  - 以上の処理を繰り返してキー値の位置を求める

2分探索の例

Step 1 データ範囲の初期化: i=0, j=7

0	1	2	3	4	5	6	7
10	20	30	40	50	60	70	80

Step 2 中央の位置を決める: m=(i+j)/2=3

0	1	2	3	4	5	6	7
10	20	30	40	50	60	70	80

2分探索の例

Step 3 キー値と中央のデータと比較する

3.1 等しければ中央位置m=3を解とし、探索終了

3.2 キー値のほうが小さければ、データの範囲を前半に絞る：i=0, j=m-1=2

0	1	2	3	4	5	6	7
10	20	30	40	50	60	70	80

3.3キー値のほうが大きければ、データの範囲を後半に絞る：i=m+1=4, j=7

0	1	2	3	4	5	6	7
10	20	30	40	50	60	70	80

Step2, 3を$i \leq j$が成立しなくなるまで繰り返す

Step2, 3の繰り返し回数（分割回数）の上限は
$\log_2 n$（$n/2^k=1$で、$k=\log_2 n$）なので、計算量は$O(\log_2 n)$となる。

2分探索アルゴリズム

入力：昇順（降順）にソートされた配列a[0],...,a[n-1], キー値x
出力：キー値xがaの中の位置p（存在しなければ-1）
補助：i, j, m

1. 初期化：i=?, j=?, p=-1
2. 条件i<=jが満たされる間、次の処理を繰り返す

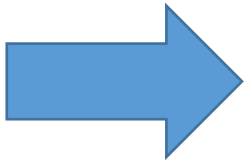
?

文字列データの場合

文字列データのソート（辞書順・大文字<小文字）

元データ

Takada
Nakata
Kobayashi
Kimura
Murata
Koyama
Akiba
Abe



ソート後のデータ

Abe
Akiba
Kimura
Kobayashi
Koyama
Murata
Nakata
Takada

文字列データの表探索（2分探索の例）

Step 1 データ範囲の初期化：i=0, j=7

0	1	2	3	4	5	6	7
Abe	Akiba	Kimura	Kobayashi	Koyama	Murata	Nakata	Takada

Step 2 中央の位置を決める：m=(i+j)/2=3

0	1	2	3	4	5	6	7
Abe	Akiba	Kimura	Kobayashi	Koyama	Murata	Nakata	Takada

文字列データの表探索（2分探索の例）

Step 3 キー値と中央のデータと比較する

3.1 等しければ中央位置mを解とし、探索終了

3.2 キー値のほうが小さければ、データの範囲を前半に絞る：i=0, j=2

0	1	2	3	4	5	6	7
Abe	Akiba	Kimura	Kobayashi	Koyama	Murata	Nakata	Takada

3.3キー値のほうが大きければ、データの範囲を後半に絞る：i=4, j=7

0	1	2	3	4	5	6	7
Abe	Akiba	Kimura	Kobayashi	Koyama	Murata	Nakata	Takada

Step2, 3を$i \leq j$が成立しなくなるまで繰り返す

文字列のプログラミング

- 変数のデータ型は？
  - 整数の場合はint xやint a[100]
- 「文字列が等しい」の判定は？
  - 数値の場合はif(x==y)やif(x==10)
- 文字列の大小比較は？
  - 数値の場合はif(x>y)など
- 文字列値の付与は？
  - 数値の場合x=100
- 文字列の入出力は？
  - 整数の場合scanf(“%d”, &x), printf(“%d¥n”, x)
- これらが分かれば、数値用のアルゴリズムを文字列用に書き換えるだけで済む

文字列について

- 1次元配列
  - 宣言：char s[64]; or char s[]=“Ryukoku”;
  - 値の付与：strcpy(s, “Ryukoku”);
  - 計算機上の格納：

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]
R	y	u	k	o	k	u	¥0

  - s: 上記文字列を格納している先頭のメモリのアドレス⇒変数へ読み込む時は、&sはなく、sを使う⇒scanf(“%s”, s);
  - ちなみに、int xの場合、&xがメモリのアドレス⇒scanf(“%d”, &x);

文字列について

- 2次元配列
  - 宣言：char s[3][64]; or char s[][64]={“Ryukoku”, “Uni”, “Seta”};
  - 値の付与：strcpy(s[0], “Ryukoku”); strcpy(s[1], “Uni”); strcpy(s[2], “Seta”);
  - 計算機上の格納：

	0	1	2	3	4	5	6	7
0	R	y	u	k	o	k	u	¥0
1	U	n	i	¥0				
2	S	e	t	a	¥0			

  - s[0], s[1], s[2]: 個々の文字列を格納している先頭のメモリのアドレス
    ⇒scanf(“%s %s %s”, ???);

文字列について

- 2次元配列
  - 宣言：char s[3][64]; or char s[][64]={“Ryukoku”, “Uni”, “Seta”};
  - 値の付与：strcpy(s[0], “Ryukoku”); strcpy(s[1], “Uni”); strcpy(s[2], “Seta”);
  - 計算機上の格納：

	0	1	2	3	4	5	6	7
0	R	y	u	k	o	k	u	¥0
1	U	n	i	¥0				
2	S	e	t	a	¥0			

- s[0], s[1], s[2]: printf(“…”, s[0], s[1], s[2]); ???
- s[1][2]: printf(“…”, s[1][2]); ???

文字列のフォーマット化標準入出力関数

関数名	使い方	戻り値	機能
scanf	#define N 64 char s[N]; scanf(“%s”, s);	成功：入力項目数 失敗：EOF	標準入力から文字列をsに読み込む
printf	#define N 64 char s[N]; printf(“%s”, s);	成功：書き出された文字の数 失敗：負の値	文字列sを書き出す

文字列処理

関数名	使い方	戻り値の型	機能
strcmp	strcmp(s, t);	int	文字列sとtを比較。s==tなら0を、s<tなら負の値を、s>tなら正の値を返す。ここでの大小は辞書順である。また、大文字<小文字
strcpy	strcpy(s, t);	char *	文字列tをsにコピーし、sを返す

- ヘッダファイル<string.h>が必要
- ライブラリ関数内では、変数s：char *, 変数t：const char *
  - ✓ main関数側であればchar s[N];　1次元配列の場合
  - ✓ 関数の引数であればchar *s, or, char s[]; 1次元配列の場合

文字列データのソートプログラム

```
void SSort(int n, int a[])
{
  int i, j, min, no;

  for(i=0; i<n-1; i++){
    min=a[i]; no=i;
    for(j=i+1; j<n; j++)
      if(min>a[j]) {min=a[j]; no=j;}
    if(no != i){
      a[no]=a[i]; a[i]=min;}
  }
}
```

```
void SSort(int n, char a[][64])
{
  int i, j, ~~min,~~no; char ????;

  for(i=0; i<n-1; i++){
    ???; no=i;
    for(j=i+1; j<n; j++)
      if(???) {???; no=j;}
    if(no != i){
      ???; ???;}
  }
}
```

第2回演習課題

- 注意：課題1,2については、課題2ができた時点で提出してください。

第2回演習課題

1. 線形探索を実現する関数int LSearch(int n, int a[], int x)を作成し、その動作を確認できるプログラム（ex02-lsearch.c）を作成しなさい。ただし、nはデータの個数、a[]はデータ配列である。

第2回演習課題

2. 2分探索を実現する関数int BSearch(int n, int a[], int x)を作成し、その動作を確認できるプログラム（ex02-bsearch.c）を作成しなさい。ただし、nはデータの個数、a[]はデータ配列である。また、探索過程において、探索範囲内のデータを示しなさい。

実行例

```
./a.out
Input the number of data: 10
Input the data
10 20 30 40 50 60 70 80 90 100
Input the search key: 60
the data to be searched
10 20 30 40 50 60 70 80 90 100
the data to be searched
60 70 80 90 100
the data to be searched
60 70
Search result: 5
```

第2回演習課題

3. 文字列データの2分探索を実現する関数int BSearch(int n, char a[][N], char x[N])を作成し、その動作を確認できるプログラム（ex02-bsearch-str.c）を作成しなさい。ただし、nはデータの個数、a[][N]は文字列（複数用）の配列である。strcmp()など文字列の標準ライブラリを使うために、ヘッダstring.hをincludeする必要がある。

実行例：

```
./a.out
Input the number of data: 8
Input the data
abe akiba kimura kobayashi koyama murata nakata takada
Input the search key: nakata
Search result: 6
```

第2回演習課題

4. 2分探索を実現する関数int BSearch(int i, int j, int a[], int x)を、再帰で作成し、その動作を確認できるプログラム（ex02-bsearch-re.c）を作成しなさい。ただし、i, jはデータ範囲となる配列の左右の要素番号、a[]はデータ配列である。また、探索過程において、探索範囲内のデータを示しなさい。発展課題

実行例

```
./a.out
Input the number of data: 10
Input the data
10 20 30 40 50 60 70 80 90 100
Input the search key: 60
the data to be searched
10 20 30 40 50 60 70 80 90 100
the data to be searched
60 70 80 90 100
the data to be searched
60 70
Search result: 5
```