

プログラミング応用

Week11

探索アルゴリズムの学習内容

- データ構造
 - 線形データ構造
 - グラフデータ構造
- 探索アルゴリズム
 - 線形探索
- 演習
 - 線形探索アルゴリズムの実装

「データ(Data)」とは

- 例) ひたちなか市の気温データ

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

- 例) ディレクトリ構造のデータ

st15dxx/ --- c-programming-practice/

-- kadai1.c

-- kadai2.c

--- 講義資料/

-- week1.pdf

-- week2.pdf

「データ(Data)」とは

- 例) ひたちなか市の気温データ

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

- 例) ディレクトリ構造のデータ

st15dxx/ --- c-programming-practice/

-- kadai1.c

-- kadai2.c

--- 講義資料/

-- week1.pdf

-- week2.pdf

データ構造

- 線形データ構造

- 気温データのように横に並べることが出来る構造

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

- グラフデータ構造

- ディレクトリ構造、家系図のように横に並べて描くことの出来ない構造

グラフデータ構造については4年生の選択科目で学習する
3年生は線形データ構造を中心に学習

C言語による線形データの表現方法

- 配列での表現が基本
- C言語による気温データの表現

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C



```
// 線形データ(日付、気温)をそれぞれ配列で表現
int dates[DATA_SIZE] = {601, 602, 603, 604, 605};
int temperatures[DATA_SIZE] = {25, 26, 30, 24, 23};
```

プログラム例(linear-data.c)

```
1 #include <stdio.h>
2
3 #define DATA_SIZE 5
4
5 int main(void) {
6     // 線形データ(日付、気温)をそれぞれ配列で表現
7     int dates[DATA_SIZE] = {601, 602, 603, 604, 605};
8     int tempretures[DATA_SIZE] = {25, 26, 30, 24, 23};
9     // 線形データを表示
10    for(int i = 0; i < DATA_SIZE; i++) {
11        printf("%d: %d\n", dates[i], tempretures[i]);
12    }
13    return 0;
14 }
```

探索アルゴリズムの学習内容

- データ構造
 - 線形データ構造
 - グラフデータ構造
- 探索アルゴリズム
 - 線形探索
- 演習
 - 線形探索アルゴリズムの実装

探索

- 探索＝条件を満たすデータを探す処理
 - － 探索は多くのソフトウェアで必要になる基盤処理
 - 検索サービス) キーワードを含むページを探索
 - グルメサイト) 指定した都道府県の飲食店を探索

探索の例

- 6月の真夏日を探索する

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

入力: 6月の気温データ(配列で与えられている)

出力: 真夏日があれば日付を出力

この例であれば6/3を出力したい

探索アルゴリズム

- 指定した条件を満たすデータを探しだすアルゴリズム
 - より効率的で、メモリ使用が少ない方が良い
 - 実用的には実装が簡単な方が良い

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

皆さんだったらどのようなアルゴリズムで6月の真夏日を探索しますか？
(世界中の研究者が未だにより良いアルゴリズムを探し続けています)

線形データの探索アルゴリズム

1. 線形探索
2. 線形探索＋番兵法
3. 二分探索(次回の内容)

線形探索（例1）

- 基本方針：
配列の先頭から条件を満たすか調べる

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

ステップ1

1. 6/1が真夏日であるか調べる
2. 6/1がデータの末尾でないか調べる
→ 末尾ではないので配列の次の要素に進む

線形探索（例1）

- 基本方針：
配列の先頭から条件を満たすか調べる

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

ステップ2

- 6/2が真夏日であるか調べる
- 6/2がデータの末尾でないか調べる
→ 末尾ではないので配列の次の要素に進む

線形探索（例1）

- 基本方針：
配列の先頭から条件を満たすか調べる

6/1	6/2	6/3	6/4	6/5
25°C	26°C	30°C	24°C	23°C

ステップ3

- 6/3が真夏日であるか調べる
→ 30°C以上なので探索終了！

線形探索（例2）

- 基本方針：
配列の先頭から条件を満たすか調べる

6/1	6/2	6/3	6/4	6/5
25°C	26°C	23°C	24°C	23°C

ステップ1

1. 6/1が真夏日であるか調べる
2. 6/1がデータの末尾でないか調べる
→ 末尾ではないので配列の次の要素に進む

線形探索(例2)

- 基本方針:
配列の先頭から条件を満たすか調べる

6/1	6/2	6/3	6/4	6/5
25°C	26°C	23°C	24°C	23°C

ステップ2

1. 6/2が真夏日であるか調べる
2. 6/2がデータの末尾でないか調べる
→ 末尾ではないので配列の次の要素に進む

線形探索(例2)

- 基本方針:
配列の先頭から条件を満たすか調べる

6/1	6/2	6/3	6/4	6/5
25°C	26°C	23°C	24°C	23°C

ステップ5

- 6/5が真夏日であるか調べる
- 6/5がデータの末尾でないか調べる
→末尾まで来たので探索終了
真夏日は見つからなかった

C言語による表現(linear-search.c)

```
1 #include <stdio.h>
2
3 #define DATA_SIZE 5
4
5 int main(void) {
6     // 線形データ(日付、気温)をそれぞれ配列で表現
7     int dates[DATA_SIZE] = {601, 602, 603, 604, 605};
8     int temperatures[DATA_SIZE] = {25, 26, 30, 24, 23};
9     // 線形データを表示
10    int i = 0;
11    while(1) {
12        if(temperatures[i] >= 30){
13            printf("%dが真夏日です。 \n", dates[i]);
14            break;
15        }
16        if (i == DATA_SIZE){
17            printf("合致するデータがありません。 \n");
18            break;
19        }
20        ++i;
21    }
22    return 0;
23 }
```

線形探索の利点・欠点

- 利点
 - もっとも単純な探索アルゴリズム
= 実装が簡単
- 欠点
 - 先頭から調べていくので条件を満たすデータが末尾の方にあると探索に時間が掛かる

6/1	6/2	6/3	...	6/30
25°C	26°C	25°C	...	30°C

6/30にたどり着くまで真夏日が見つからない。
→ この欠点を解決する二分探索というアルゴリズムを来週学習します

線形探索の改善

- linear-search.cをさらに高速化することは可能か？
 - 番兵法という高速化手法が有名

番兵法による高速化

- 番兵法：配列の末尾に条件を満たすデータを追加することで高速化する手法
 - 6月の真夏日を探索する場合ならば、データの末尾に30°Cのデータを挿入する
(日付はダミーの0などで良い)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	30°C	24°C	23°C	30°C

番兵法の動作(真夏日がある場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	30°C	24°C	23°C	30°C

ステップ1

1. 6/1が真夏日であるか調べる
 2. ~~6/1がデータの末尾でないか調べる~~
- 番兵法ではこの比較が不要！

番兵法の動作(真夏日がある場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	30°C	24°C	23°C	30°C

ステップ2

1. 6/2が真夏日であるか調べる
 2. ~~6/2がデータの末尾でないか調べる~~
- 番兵法ではこの比較が不要！

番兵法の動作(真夏日がある場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	30°C	24°C	23°C	30°C

ステップ2

1. 6/3が真夏日であるか調べる
→ 6/3は真夏日
2. 6/3がデータの末尾でないか調べる
→ 末尾ではないので6/3は真夏日

番兵法の動作(真夏日がある場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	30°C	24°C	23°C	30°C

ステップ2

1. 6/3が真夏日であるか調べる
→ 6/3は真夏日
2. 6/3がデータの末尾でないか調べる
→ 末尾ではないので6/3は真夏日

番兵法は条件にヒットするデータを見つけたときのみ
末尾であるか調べるので速い

番兵法の動作(真夏日がない場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	20°C	24°C	23°C	30°C

ステップ1

1. 6/1が真夏日であるか調べる
 2. ~~6/1がデータの末尾でないか調べる~~
- 番兵法ではこの比較が不要！

番兵法の動作(真夏日がない場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	20°C	24°C	23°C	30°C

ステップ2

1. 6/2が真夏日であるか調べる
 2. ~~6/2がデータの末尾でないか調べる~~
- 番兵法ではこの比較が不要！

番兵法の動作(真夏日がない場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	20°C	24°C	23°C	30°C

ステップ3

1. 6/3が真夏日であるか調べる
2. ~~6/3がデータの末尾でないか調べる~~
番兵法ではこの比較が不要！

番兵法の動作(真夏日がない場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	20°C	24°C	23°C	30°C

ステップ4

1. 6/4を調べる
2. ~~6/4がデータの末尾か調べる~~

番兵法の動作(真夏日がない場合)

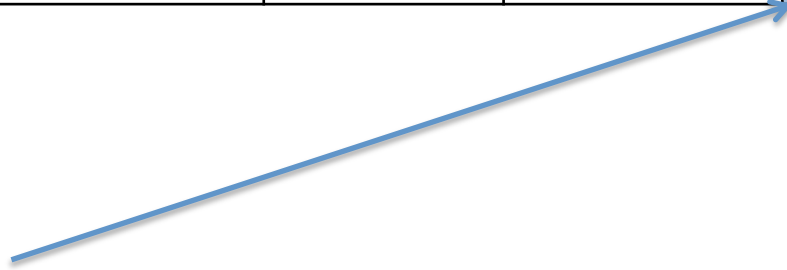
6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	20°C	24°C	23°C	30°C

ステップ5

1. 6/5が真夏日か
2. ~~6/5が末尾か~~

番兵法の動作(真夏日がない場合)

6/1	6/2	6/3	6/4	6/5	0
25°C	26°C	20°C	24°C	23°C	30°C



ステップ6

1. 0を調べる
→ 30°Cなので真夏日！
2. データの末尾か調べる
→ 番兵法は条件にヒットしたらデータの末尾か否か調べる
→ データの末尾はダミーなので「真夏日なし」

サンプルプログラム

(linear-search-banpei.c)

```
1 #include <stdio.h>
2
3 #define DATA_SIZE 5
4
5 int main(void) {
6     // 線形データ(日付、気温)をそれぞれ配列で表現
7     // 末尾に真夏日のデータを挿入
8     int dates[DATA_SIZE+1] = {601, 602, 603, 604, 605, 0};
9     int temperatures[DATA_SIZE+1] = {25, 26, 29, 24, 23, 30};
10    // 線形データを表示
11    int i = 0;
12    while(1) {
13        if(temperatures[i] >= 30){
14            if (dates[i] != 0) {
15                printf("%dが真夏日です。\\n", dates[i]);
16            } else {
17                printf("合致するデータはありません。");
18            }
19            break;
20        }
21        ++i;
22    }
23    return 0;
24 }
```

探索アルゴリズムの学習内容

- データ構造
 - 線形データ構造
 - グラフデータ構造
- 探索アルゴリズム
 - 線形探索
- 演習
 - 線形探索アルゴリズムの実装

演習0

1. week11ディレクトリを作成しなさい
2. 講義資料中の以下のプログラムを記述し動作を確認しなさい。
 1. linear-data.c
 2. linear-search.c
 3. linear-search-banpei.c

演習1

- 以下の線形データを考える。

出席番号	1	2	3	4
得点	60	65	33	90

- 以下の動作をするプログラムkadai1.cというファイル名で保存し提出しなさい
 - 出席番号、得点のデータを配列で表現
 - scanf()関数を用いて得点をキーボードから入力
 - 入力した得点を獲得した学生を線形探索で探し、見つければ「出席番号xxがyy点でした。」と表示

演習2

- 演習1で作成したプログラムを番兵法で高速化しなさい
(banpei.cというファイル名で保存)

次回

- 二分探索木と呼ばれるデータ構造からデータを探索するアルゴリズムを学習します
- “計算量”の考え方を学習します