ななちゃんのIT教室

読まなくて良い教養講座の巻

by nara.yasuhiro@gmail.com

ななちゃんが
アルゴリズムを勉強するという お話

第 0.3 版　2017 年 7 月 7 日

フリー素材
http://freeillustration.net

もくじ

第1回　アルゴリズムは教養

なな：この副教材は読まなくて良いの？

先生：読まなくても、直ちに生命に危険が及ぶということはありません。教養ってそういうものです。教養とは、社会人として必要な広い文化的な知識、または、それによって養われた品位のこと。単なる知識ではなく、人間がその素質を精神的、全人的に開化、発展させるために、学び養われる学問や芸術などのこと。英語の culture、ドイツ語の Bildung に対する訳語なの。

なな：culture って、文化のことじゃないの？

先生：英語の culture は、もともと、耕作･養育の意味、ドイツ語の Bildung は、形成･教化の意味なの。cultivate は、耕す、栽培する、養殖する、培養する、養う、磨く、洗練する、修めるという意味でしょ。農業の agriculture は、ラテン語の、ager（畑）＋ cultura（耕作する）の agricultura が語源だし。Bildung は、英語の build と同じ語源。

なな：それが、プログラミングとどういう関係があるの？

先生：**プログラミングの場合、典型的な教養は、配列データの合計、最大値、検索（サーチ）、並べ替え（ソート）**などです。こういうのを、一般に「**アルゴリズム**」といいます。実際には、そういう機能は、ライブラリ（呼び出すだけで使える関数）として用意されていることが多いので、それを利用すれば良いの。原理を知っている必要は、必ずしも必要ありません。たとえば、JavaScript には sort 関数があります。

```
var numbers = [ 4, 2, 5, 1, 3 ] ;
numbers.sort ( function (a, b) { return a - b; } ) ;
alert(numbers);
```

大小関係の指定
b – a にすれば 大→小

1、2、3、4、5 という
アラート表示になる

例えで言えば、 電子レンジの原理を知らなくても、 電子レンジを買ってきて、 ボタンをチンすれば困らない。だから、この資料は読まなくて良い。

この資料の内容を知らなくても困らないけど、**知っていれば、プログラマとしての人間に厚みができる**。

ちなみに、「アルゴリズム」というのは、問題を解く方法。算法と訳されます。アルゴリズムの教科書には、たいがい、検索（サーチ）、並べ替え（ソート）が出てくる。本当は、もっと広い概念で、脳細胞の仕組みを参考にしたニューラルネットワーク、突然変異や遺伝進化を参考にした遺伝的アルゴリズムなんかも含まれます。

なな：「知っていれば、プログラマとしての人間に厚みができる」とか、自尊心を刺激して、読ませようという魂胆が丸見え。

先生：ばれたか。

第2回 アルゴリズム初級編

先生：ななちゃん、「var x=2, y=3;」というデータがあるとして、x と y の**データを交換**するにはどうすれば良いかしら？

なな：こうかな？

```
var x=2, y=3;
x = y;
y = x;
alert ( x + ", " + y );
```

あれれ、「3, 3」と表示される！

先生：正解はこれ。

```
var x=2, y=3;
var tmp = x;
x = y;
y = tmp;
alert(x + ", " + y);
```

なな：変数をひとつ余計に用意するのね。ちょっとずるい。

先生：こういう、常識にとらわれない、柔軟な発想が必要ということね。次に、**データの合計を求める**方法を説明しましょう。「sum += data [i];」は、「sum = sum + data [i];」の略記法で、「足し込む」という意味です。クラスの成績表の例です。「成績の合計を計算して何になるの？」と言われそうなので、平均値の計算に利用してみました。

```
var data = [ 58,80,59,32,51,78,67,35,48,72,27,25,33,20,88,71,
             82,36,14,50,45,19,29,39,30,61,72,41,83,31,84,44 ] ;
var sum=0;
for (var i=0; i<data.length; i++) {
    sum += data [ i ];
}
document.write ( "sum=" + sum + ”<br>” ) ;
var ave = sum / data.length;
document.write ( "ave=" + ave ) ;
```

なな：先生、被害妄想よ。

先生：次に、**最大値を求めるプログラム**です。「if (max < data[i]) max = data[i];」は、暫定的な最大値より大きいデータがあったら、暫定的な最大値をそれに更新する、という意味です。

```
var data = [ 58,80,59,32,51,78,67,35,48,72,27,25,33,20,88,71,
             82,36,14,50,45,19,29,39,30,61,72,41,83,31,84,44 ] ;
var max=0;
for (var i = 0; i < data.length; i++) {
    if (max < data [ i ]) max = data [ i ] ;
}
document.write( "max=" + max ) ;
```

第3回 並べ替え（ソート）

先生：順不同のデータを、小さい数→大きい数の順などに**並べ替え**ることを 「**ソート**」 といいます。簡単な実現方法は、最小値をみつけて先頭に置き、残りの中から最小値を見つけて 2 番目に置く、というようなことを繰り返す方法です。選択ソートと呼ばれています。

```
var data = [ 58,80,59,32,51,78,67,35,48,72,27,25,33,20,88,71,
             82,36,14,50,45,19,29,39,30,61,72,41,83,31,84,44 ] ;
for ( var i = 0; i < data.length; i++ ) {
    var mn = 100, ix;
    for (var j = i; j < data.length; j++) {
        if ( mn>data [ j ] ) { mn = data [ j ]; ix = j; }
    }
    data [ ix ] = data [ i ];
    data [ i ] = mn;
}
document.write ( JSON.stringify(data) ) ;
```

最小値更新時の j を記憶

最小値を先頭と交換

ソート後のデータを表示

データの個数が、何万、何十万になると、計算時間がかかるので、計算量を減らす工夫が研究されています。

1960年にアントニー・ホーアが開発したクイックソート。基本的な考え方は

1. 適当な数（ピボットという）を選択する （データの総数の中央値が望ましい）
2. ピボットより小さい数を前方、大きい数を後方に移動させる （分割）
3. 二分割された各々のデータを、それぞれソートする

実際にこれを実現するためのアルゴリズムは色々考えられます。たとえば、以下のようなものがあります。

1. ピボットとして一つ選びそれをPとする。
2. 左から順に値を調べ、P 以上のものを見つけたらその位置をiとする。
3. 右から順に値を調べ、P 以下のものを見つけたらその位置をjとする。
4. I が j より左にあるのならばその二つの位置を入れ替え、2 に戻る。
   ただし、次の2での探索はiの一つ右、次の 3 での探索は j の一つ左から行う。
5. 2 に戻らなかった場合、i の左側を境界に分割を行って 2 つの領域に分け、
   そのそれぞれに対して再帰的に1からの手順を行う。
   要素数が 1 以下の領域ができた場合、その領域は確定とする。

```
var data = [58,80,59,32,51,78,67,35,48,72,27,25,33,20,88,71,
            82,36,14,50,45,19,29,39,30,61,72,41,83,31,84,44];
q_sort (data, 0, data.length-1);
document.write ( JSON.stringify(data) ) ;

function q_sort (numbers, left, right) {
    var pivot, l_hold, r_hold;
    l_hold = left;
    r_hold = right;
    pivot = numbers[left];
    while (left < right) {
        while ((numbers[right] >= pivot) && (left < right)) right--;
        if (left != right) {
            numbers[left] = numbers[right]; left++;
        }
        while ((numbers[left] <= pivot) && (left < right)) left++;
        if (left != right) {
            numbers[right] = numbers[left]; right--;
    }   }
    numbers[left] = pivot;
    if (l_hold < left) q_sort(numbers, l_hold, left-1);
    if (r_hold > left) q_sort(numbers, left+1, r_hold);
}
```

ソート後のデータを表示

第4回 検索（サーチ）

先生：多数のデータの中から、**特定のデータをみつける**処理を「**検索**」（**サーチ**）と呼びます。

```
var data = [58,80,59,32,51,78,67,35,48,72,27,25,33,20,88,71,
            82,36,14,50,45,19,29,39,30,61,72,41,83,31,84,44];
var cpr=0, key=84;
for (var i=0; i<data.length; i++) {
    cpr++;
    if (data[ i ] == key) break;
}
document.write("idx=" + I + "<br>");
document.write("cpr=" + cpr);
```

何番目のデータだったか表示

比較が何回必要だったか表示

何回比較すればみつかるかな？

なな：探すデータが、先頭にあることもあるし、末尾にあることもあるから、平均すれば、データ個数の半分ね。

先生：そうね。 その計算量を減らすための工夫が、データを前もって、ソートしておくことなの。

なな：ソートすること自体に計算時間がかかるわね。

先生：ソートを 1 回やっておけばいいから、何回も検索をやる場合に有利ね。例えば、みんなが計算機を使わないような真夜中にソートをしておけば、翌日は高速に検索できるという使い方もできます。

```
var data = [58,80,59,32,51,78,67,35,48,72,27,25,33,20,88,71,
            82,36,14,50,45,19,29,39,30,61,72,41,83,31,84,44];
data.sort()sort(function(a, b) { return a - b; });
var cpr=0, key=84;
var fm=0, to=data.length-1;
for (;;) {
    cpr++;
    var ix = Math.floor((fm+to)/2);
    if (data[ix]==key) break;
    if (data[ix]<key) fm = ix+1;
    else              to = ix-1;
}
document.write(JSON.stringify(data)+"<br>");
document.write("idx="+ix+"<br>");
document.write("cpr="+cpr);
```

JavaScript 組み込みのソート関数

みつかった

次は後半を検索

次は前半を検索

まず、真ん中のデータを調べて、対象が前半にあるか、後半にあるかを決定し、その範囲だけを検索します。このようなことを繰り返せば、範囲が、1/2、1/4、1/8、1/16、と、どんどん範囲が狭まります。こういうサーチを「**バイナリサーチ**」（**二分検索**）といいます。

「data.sort()sort(function(a, b) { return a - b; });」は、JavaScript の配列データ用のソート関数です。「function(a, b) { return a - b; }」は、大小関係を決める方法の指定です。「a – b」を「b - a」にすれば、大→小 の方向に並べることができます。

第5回　応用（84点とったのは誰？

なな：成績データの順番を変えてから検索したら、見つかったデータが誰の成績だったか分からなくなってしまうのでは？

先生：そういう場合は、ひとつのデータを「［学籍番号，成績］」のように、ペアにすれば良いのよ。

学籍番号　　成績

```
var data = [ [1,58],[2,80],[3,59],[4,32],[5,51],[6,78],[7,67],[8,35],[9,48],
             [10,72],[11,27],[12,25],[13,33],[14,20],[15,88],[16,71],[17,82],
             [18,36],[19,14],[20,50],[21,45],[22,19],[23,29],[24,39],[25,30],
             [26,61],[27,72],[28,41],[29,83],[30,31],[31,84],[32,44] ];
document.write ( JSON.stringify(data) + "<br>" ) ;

q_sort2(data,0,data.length-1);
document.write ( JSON.stringify(data) + “<br>” ) ;

var cpr=0, key=84;
var fm=0, to=data.length-1;

for (;;) {
    cpr++;
    var ix = Math.floor((fm+to)/2);
    if (data [ix] [1] == key) break;
    if (data [ix] [1] <   key) fm = ix+1;
    else                       to = ix-1;
}

document.write("id=" + data [ix] [0] + "<br>");
document.write("cpr=" + cpr + "<br>");

function q_sort2(numbers, left, right) {
      var pivot, l_hold, r_hold;
      l_hold = left;
      r_hold = right;
      pivot = numbers[left];
      while (left < right) {
            while ((numbers [right] [1] >= pivot [1]) && (left < right)) right--;
            if (left != right) {
                  numbers [left] = numbers[right];
                  left++;
            }
            while ((numbers [left] [1] <= pivot [1]) && (left < right)) left++;
            if (left != right) {
                  numbers [right] = numbers [left];
                  right--;
            }
      }
      numbers [left] = pivot;
      if (l_hold < left) q_sort2 (numbers, l_hold, left-1);
      if (r_hold > left) q_sort2 (numbers, left+1, r_hold);
}
```

ソート前データを表示

ソート後データを表示

みつかったデータの学籍番号を表示

ソートに、JavaScript 組み込みの関数を使う場合は、下記のようにします。

```
data.sort ( function (a, b) { return a [1] – b [1] ; } ) ;
```

成績を比較