提出日　令和2年07月09日

アルゴリズムとデータ構造　第6回課題レポート

学籍番号（A19117）
氏名（永尾優磨）

【課題】

1. 今回のソート方法2種類と、それ以外のもう1種類のソート方法について調査し、説明せよ。
2. バブルソートについて以下の作業を行い、ソースリストと実行結果をレポートにまとめよ。
   - バブルソートクラスとそれを実行するクラスを作成せよ。
   - 適当なデータを放り込み（10個程度）、バブルソートが行えるかどうか確認せよ。

3. クイックソートについても以下の作業を行い、ソースリストと実行結果をレポートにまとめよ。
   - クイックソートクラスとそれを実行するクラスを作成せよ。
   - 適当なデータを放り込み（10個程度）、クイックソートが行えるかどうか確認せよ。

【作成したプログラム】

```
public class Bubble {

    private int[] a;
    private int nElems;

    Bubble(int max) {
        a = new int[max];
        nElems = 0;
    }

    public void insert(int value) {
        a[nElems++] = value;
    }

    public void display() {
        int i;
        for (i = 0; i < nElems; i++) {
            System.out.println(a[i]);

        }
    }

    public void bubbleSort() {
```

```
        int out;
        int in;
        display();
        System.out.println("--------------");
        for (out = nElems - 1; out > 0; out--) {
            for (in = 0; in < out; in++) {
                if (a[in] > a[in + 1]) {
                    int tmp;
                    tmp = a[in];
                    a[in] = a[in + 1];
                    a[in + 1] = tmp;
                }
            }
            display();
            System.out.println("--------------");

        }
    }

}
```

プログラム1.1　Bubble

```
import java.util.Random;

public class BubbleApp {
    public static void main(String[] args) {
         int n = 5;
        Bubble app = new Bubble(n);
        Random rnd = new Random();
        for (int i = 0; i < n - 1; i++) {
            app.insert(rnd.nextInt());
        }
        app.bubbleSort();
    }

}
```

プログラム1.2　BubbleApp

```
public class Quick {
    private int[] a;
    private int nElems;

    Quick(int max) {
        a = new int[max];
        nElems = 0;
    }
```

```
public void insert(int value) {
    a[nElems++] = value;
}

public void display(int l, int r) {
    int i;
    for (i = 0; i < nElems; i++) {
        System.out.print(a[i] + ",");
    }
    System.out.print("l=" + l + ",r=" + r + ",");

}

public void quickSort(int left, int right) {
    int pivot;
    int i;
    int j;
    System.out.println(left + "から" + right + "までソート");
    pivot = a[(left + right) / 2];

    display(left, right);
    System.out.println("pivot = " + pivot);
    i = left;
    j = right;
    for (;;) {
        while (a[i] < pivot) {
            i++;
        }
        while (pivot < a[j]) {
            j--;
        }
        if (i >= j) {
            break;
        }
        int tmp = a[i];
        a[i] = a[j];
        a[j] = tmp;

        i++;
        j--;
    }
    if (left < i - 1) {
        quickSort(left, i - 1);
    }
    if (j + 1 < right) {
        quickSort(j + 1, right);
    }
```

```
    }

}
```

プログラム2.1　Quick

```
import java.util.Random;

public class QuickApp {
    public static void main(String[] args) {
        int n = 15;
        Quick app = new Quick(n);
        Random rnd = new Random();

        for (int i = 0; i < n; i++) {
            app.insert(rnd.nextInt());
        }

        app.display(0, n);

        app.quickSort(0, n - 1);
        System.out.println("---------");
        app.display(0, n - 1);
    }
}
```

プログラム2.2　QuickApp

【プログラムの解説】

Bubbleのdisplayの関数は、授業でのスライドから内容を変更した。シンプルに、1行ずつ文を出力する形に変更し、見やすくするために、プリント文の間に---------と出力する文章を付け加えた。

QuickAppは、Randomなどを用いて、データの生成を容易に出来るようにした。

【結果】

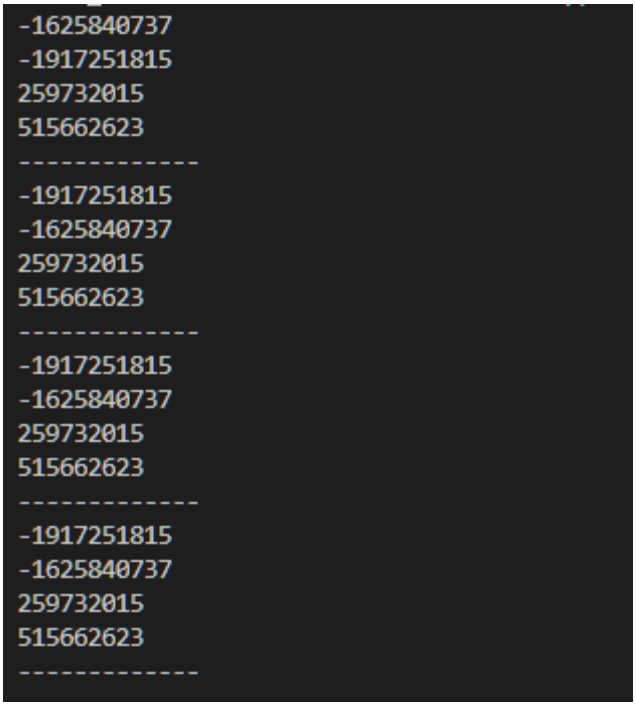

図1　BubbleAppの出力結果

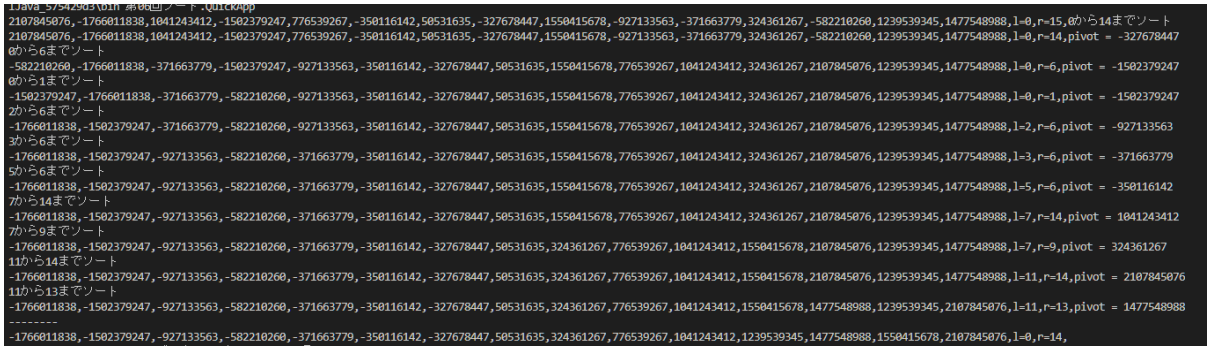

図2　QuickAppの出力結果

【課題1】

今回のソート方法2種類と、それ以外のもう1種類のソート方法について調査し、説明せよ。

・バブルソート

リストのとなったデータを比較して、大小の順序が違っているときは並べ替えていく方法です。データが移動していく様子を、水中で泡が浮かんでいく様子に例えて、バブルソートという名前が付けられいる。

リストの先頭と次のデータから初めて、左の方が大着れば右と交換することを、1つずつずらしながら繰り返す。リストの最後尾まで到達すると、1回目の比較は終了だ。このとき、リストの最後尾にはデータの最大値が入りる

2回目は一番右端を除いて同様の比較を行うと、最後から2番目が決まる。これをくり貸すと、全てが並べ替えられ、ソートは終了だ。（図3.1参照）

改良するとしたら

バブルソートで変更が発生しなかった場合に処理を打ち切り、高速化することを考えれる。処理中に好感が多かったかどうかを記録し、降雨感が起こらなかった場合はそれ以降の処理を行わないモノとする。

図3.1 バブルソート

・クイックソート

クイックソートはリストから出来とうにデータを1つ選んで、これを基準として小さい要素と大木要素の分割し、それぞれのリストでまた同じような処理を繰り返してソートする方法だ。

一般的には「分割当時法」とも飛ばれる方法に分類されるソート方法で、小さい単位に分割して処理することを再帰的に繰り返す。これ以上分けられないようなサイズ　まで分割できれば、それをまとめた結果を求める。

一般的には「分割統治法」とも呼ばれる方法に分類されるソート方法で、小さい単位に分割して処理することを再帰的に繰り返す。これ以上分けられないような細部まで分割できれば、それをまとめた結果を求める方法だ。（図3.2参照）

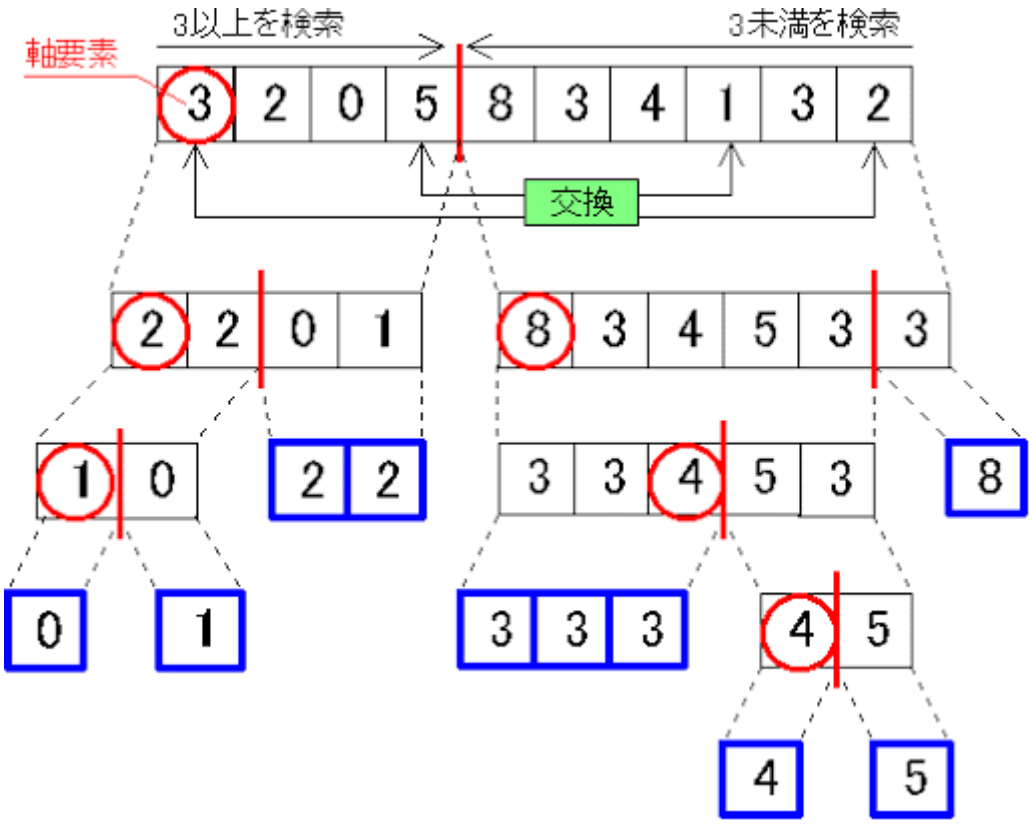

図3.2　クイックソート

・マージソート

マージソートは、ソートしたいデータが入ったリストを2つに分割することを繰り返し、全てがバラバラ担った状態から、これらのリストを統合(マージ)する方法だ。この統合する際に、そのリスト内で値が小さい順に並ぶように実装することで、全体が1つのリストになったときには全ての値がソート済みになっている。（図3.3参照）

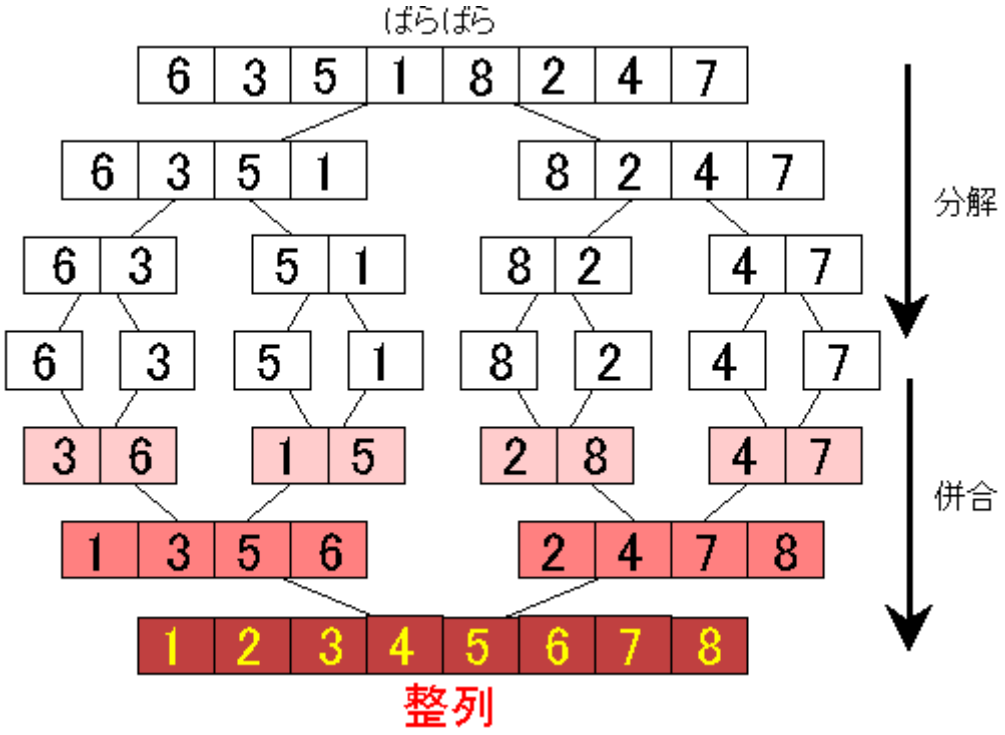

図3.3 マージソート

【考察】

数字を並べ替える時に真っ先に思いつくのが、Bubbleソートのやり方だったので、今回のQuickソートのやり方を知ることが出来て良かったと思う。裏で、Pythonを用いて、選択ソートやバブルソート、ヒープソート、クイックソートなどを動かしてみたが、クイックソートが一番速いアルゴリズムだった。これからプログラムを書く時には、ライブラリを使うので、どのようなアルゴリズムが動いている事を考えることは無いと思う。今回得た再帰を使ったソートの考え方をすぐにでも使えるようにして、これからのプログラムを書いていきたいと思う。

【参考文献】

1. 「アルゴリズムとデータ解析の授業スライド、線形探索について」
2. 「No enclosing instance of type Hoge is accessible.」〈https://qiita.com/watanabk/items/738988fac29e1e1d8d88〉
3. 増井敏克(2020)「Pythonで始めるアルゴリズム入門」翔泳社
4. 「参考文献の書き方」〈http://www7a.biglobe.ne.jp/nifongo/ron/ron_04.html〉