社工情報処理実習 1
課題 9 (テーマ: 配列とメソッド)

安藤 和敏
筑波大学社会工学系

2000 年 6 月 13 日

以下の 1 を解きなさい。さらに、2 か 3 か 4 のうち一つを選択しなさい。プログラム中にはできるだけ何をやっているのかが分かりやすいように日常言語 (日本語か英語) でコメント文 (教科書 p. 76 を参照) を入れなさい。

1. 行列の計算 [必修]

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \; B = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

とする。以下のプログラムは、これら二つの行列の和 $A + B$ を計算しその結果を出力するプログラムである。このプログラムでは、

- 二つの行列 `X` と `Y` が引数として与えられたときに、それらの和を戻り値とするメソッド `sum (double[][] X, double[][] Y)`、及び、
- 行列 X が引数として与えられたときに、それを画面に表示するメソッド `printMatrix(double[][] X)`

が定義されている。

```
public class matrix {
    public static void main (String[] args) {

        double[][] A,B;

        A = new double[][] { {1,2,3}, {4,5,6}, {7,8,9} };
        B = new double[][] { {1,4,7}, {2,5,8}, {3,6,9} };

        printMatrix(sum(A,B));

    }

    public static double[][] sum (double[][] X, double[][] Y) {
```

```
        double[][] Z = new double[3][3];

        for (int i=0; i<3; i++) {
            for (int j=0; j<3; j++) {
                Z[i][j] = X[i][j]+Y[i][j];
            }
        }

        return Z;
    }


    public static void printMatrix (double[][] X) {

          // 問題 9-9 を参照

    }
}
```

課題は上のプログラムのまねをして,

- 二つの 3×3 行列 X と Y が引数として与えられたときに、それらの積 XY を、戻り値とするメソッド product (double[] X, double[][] Y) を定義して、
- そのメソッドを呼び出すことによって, AB を計算して、
- その結果を printMatrix メソッドを使って表示する。

ようなプログラムを書くこと。

実行結果は,

```
banana:~/java% java matrix
14.0 32.0 50.0
32.0 77.0 122.0
50.0 122.0 194.0
```

のようになる。(数字が合っていれば良い。)

知っていると思いますが、行列 AB の第 (i, j) 成分は、

$$a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j}$$

です。

余力があれば、product メソッドを使って, A^3B^2 を計算するには、どうすれば良いのかを考えてみなさい。

2. コイン投げ (続き) [選択]

numOfheads(N) メソッドは、コインを N = 100 回投げたときに、表の出た回数を数えて、それを返すものであった。これは、実は二項分布に従う乱数である。(その範囲は 0 ~ 100.) もちろん、二項分布というものを知らなくてもかまいません。ただ、0 以上 100 以下の乱数ということを頭に入れてください。

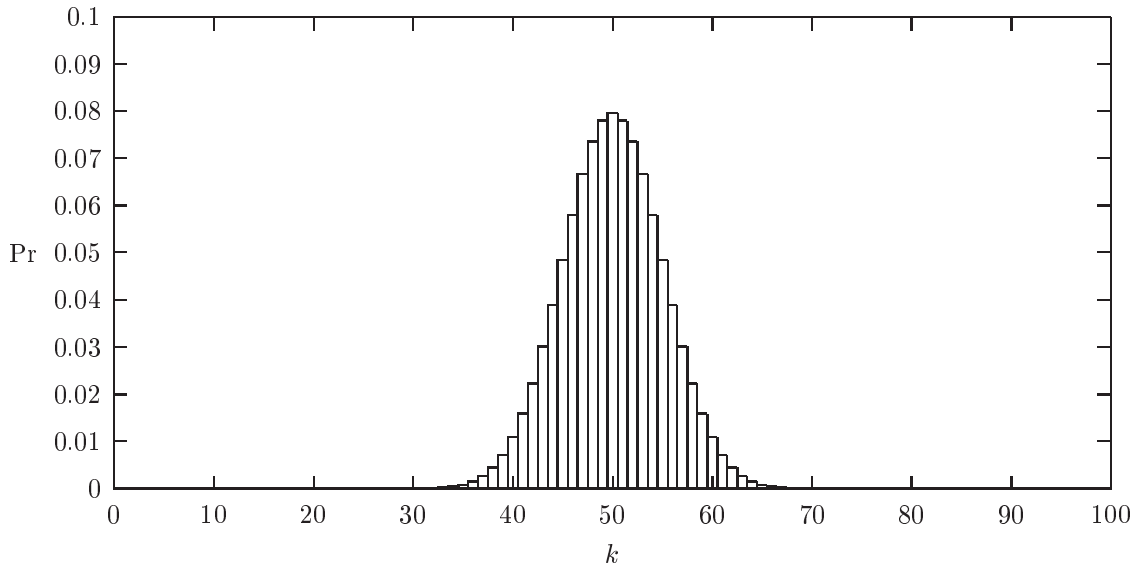

図 2.1: 二項分布

今回の課題は、この乱数を 500 個取り出してその度数分布を表すヒストグラムを書くことである。

度数を数えるためには、配列を用いると便利である。ここでは、frequency という大きさ 101 の配列を以下のように用意する。

```
int frequency[];
frequency = new int[101];
```

(ちなみに、度数のことを英語で frequency という。) それで、frequency[0] には、0 の出た回数、frequency[1] には 1 の出た回数, ···, frequency[100] には 100 の出た回数を入れるようにします。

さて、numOfheads(N) の戻り値に応じて, 度数を数えるにはどうしたら良いでしょうか？

```
int heads = numOfheads(100);  // 乱数を発生させて
if (heads == 0) {             // もし 0 が出たら
      frequency[0]++;         // frequency[0] に 1 を足す
} else if (heads == 1) {      // もし 1 が出たら
      frequency[1]++;         // frequency[1] に 1 を足す
} else if (heads == 2) {      // もし 2 が出たら
  ....
```

なんて書いていたら大変なので、

```
int heads = numOfheads(100);    // 乱数を発生させて
frequency[heads]++;             // その度数をカウント
```

と書きましょう。これを for 文を使って、500 回繰り返します。

度数が入った配列 frequency ができあがったら、最後に for 文を使って frequency[i] の個数だけ、'*' を書いて行きます。この部分は、教科書 190 ページの List 8-3 にある、printGraph メソッドを呼び出すことにしましょう。

上で述べたことを参考にして、下の不完全なプログラムを完成させて下さい。

```
public class coin3 {
```

```
    public static int numOfheads (int N) {  // 先週と同じもの

    }

    public static void printGraph (int n) { // p.190 List 8-3 を参照

    }

    public static void main (String[] args) { // main メソッドの定義

        int frequency[];
        frequency = new int[101];

        for (int k=0; k<500; k++) {         // 以下を 500 回繰り返す

            ??????????????????????????      // 乱数を発生させて

            ??????????????????????????      // その度数をカウント

        }

        ??????????????????????????????      // ヒストグラムを書く。
        ??????????????????????????????
        ??????????????????????????????

    }                                         // main メソッドの定義終り
}
```

実行結果は、以下のようになるはずである。(出力の行数が長いので、途中の行だけを示した。)

```
banana:~/java% java coin3


*


*
*
****
*******
**********
*****
**********
*********
**********
**********
**
```

```
******************************
******************************
*********************************
************************************
***************************************
*****************************************
**********************
*********************************
*******************************
**********************
*******************
*****************
***********
************
******
***


*
*
```

ちなみに本当の二項分布は、図 2のようなグラフで示されるものである。

3. 電子辞書 (数学が嫌いな人向け) [選択]

例えば、tango という二次元の配列を以下のように初期化

```
String tango[][] = {
    {"document", "文書"},
    {"bird", "鳥"},
    {"human", "人間"},
    {"face", "顔"},
    {"array", "配列"},
    {"water", "水"},
    {"good", "良い"}
};
```

すると、tango[0][0]="document", tango[0][1]="文書", tango[1][0]="bird", tango[1][1]="鳥", ... のように、この配列には文字列のデータが入る。

課題は、このような文字列配列を用いて、(非常に少ない単語数しかない) 和英電子辞書を作ることである。いきなり全部をやることは大変なのでヒントとして、以下にプログラムの例の一部を示した。

```
String line = reader.readLine();        // 英単語を読み込み
```

```
                                        // line 変数に代入する。

boolean found = false;              /* 単語が見付かれば true にする
                                       変数 */

int i=0;                            // 配列の添字

while (!found && i < N) {           /* 単語が見付かるか、あるいは、
                                       配列の最後に到達するまで
                                       以下を繰り返す。*/


    if (??????????????????) {      /* もし line 変数と等しい単語が
                                       見付かったならば , */

        ??????????????????          // 対応する単語の和訳を出力
        ??????????????????          // found を true にする
    }

    i++;                            // 添字を一つ大きくする

}                                   // while の終り

if (!found) {                       // 見付からなかったら、
    System.out.println("該当する単語は見付かりませんでした。");
}
```

line 変数に入っている文字列と ”別の文字列” を比較するためには、line.equals("別の文字列") というメソッドを使う。このメソッドの戻り値は、line=”別の文字列” か line≠”別の文字列” であるかに応じて、 true か false である。

実行結果は以下のようになる。

```
banana:~/java% java dict
face
顔
banana:~/java% java dict
school
該当する単語は見付かりませんでした。
```

余力があれば、コマンドライン引数で検索文字列を与えるプログラムや、さらに、
コマンドライン引数で、英和と和英を指定できるプログラムなどの応用を考えよ。

4. 自由課題 [選択]

今日まで習ったことを使って、何か面白いプログラム、 あるいは、自分らしいプログラムを作れ。 オリジナリティがあり、かつ、技術的にも高度なものを評価する。

あるまとまった処理を一つのメソッドで置き換えることによってプログラムが単純化されるものは、メソッドを使うことが望ましい。

自分のプログラムで「やりたいことがあるのにうまく行かない」という人は実習時間中に質問してください。それから、

http://infoshako.sk.tsukuba.ac.jp/~shakoj13/origin.html

や

http://infoshako.sk.tsukuba.ac.jp/~shakoj13/guide.html

も参考にしても良い。