提出日　令和2年07月29日

アルゴリズムとデータ構造　第9回課題レポート

学籍番号（A19117）
氏名（永尾優磨）

【課題】

1. 以下の事項について調べよ。

1.1. モンテカルロ法

数値計算手法の一つで、乱数を用いた試行を繰り返す事により近似解を求める手法。ある事象をモデル化した数式や関数があるとき、その定義域に含まれる値をランダムにたくさん生成して実際に計算をおこない、　得られた結果を統計的に処理することで推定値を得ることが出来る。数式を解析的に解くのが困難あるいは不可能な場合でも数値的に近似解を求めることが出来る。

1.2. ユークリッドの互除法

ユークリッドの互除法とは、二つの自然数の最大公約数を求めるアルゴリズムの一つ。二数の最大公約数は両者とも割り切る事が出来る自然数（公約数）のうち最大の者だが、これは　大きい方を小さい方で割ったあまりと小さい方との最大公約数に等しいという性質があり、これを利用して効率的に算出する。

1.3. エラトステネスのふるい

エラトステネスのふるいの計算量は、$O(n \log\log n)$だ。これの証明と一般的なエラトステネスのふるいの流れを記述する。

1. 2からnまでの整数を並べる。
2. 生き残っている数の中で一番小さく、まだpとして使われていないものを新たにpとおき、p以外のpの倍数を全て消す。
3. 2の操作を繰り返していき、ｐが√nを超えたら終了
4. 3の状態でふるいにかけられて残っているものが素数

2 でふるいおとすのに$\frac{n}{2}$ 回，3 でふるい落とすのに$\frac{n}{3}$回，5 でふるい落とすのに$\frac{n}{5}$回，… の演算を行うので計算量は，

$$\sum_{p<\sqrt{n}} \frac{n}{p} = \frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \cdots$$

（ただし，和は $\sqrt{n}$ 以下の素数全体に関して取る）

ここで n が十分大きいとき$\sqrt{n}$までの素数の逆数和はおおよそ

$\log\log\sqrt{n} = \log\log n + \log\frac{1}{2}$くらいなので，計算量は$O(n \log\log n)$となる。

2. 課題1~3の作業を行え。

1. モンテカルロ法により、円周率πの近似値を計算せよ。結果はπの近似値のみではなく、プロットした点の総数および円内の点の総数を併せて表示せよ。
2. 以下の2数の最大公約数を求めよ。

   7047741　2658273
3. エネトステネスのふるいを使って1000までの素数をすべて求めよ。

【作成したプログラム】

```
import java.util.Random;

public class MonteCarlo {
    public static void main(String args[]) {
        double count = 0;
        double x, y;
        double myPI;
        int k = 1000000000;
        for (int i = 0; i < k; i++) {
            x = Math.random();
            y = Math.random();

            if (x * x + y * y < 1) {
                count++;
            }
        }
        System.out.println("プロット数=" + k);
        System.out.println("円内数=" + count);

        myPI = count * 4.0 / k;
        System.out.println("PI=" + myPI);
}
```

プログラム1　MonteCarlo

```
import java.io.*;

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

        try {
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
            String[] str = bufferedReader.readLine().split(" ");
            int x = Integer.parseInt(str[0]);
            int y = Integer.parseInt(str[1]);

            System.out.println(getCommonDivisor(x, y));

        } catch (Exception e) {
            System.out.println(e);
```

```
        }
    }

    private static int getCommonDivisor(int x, int y) {
        int biggerNum = Math.max(x, y);
        int smallerNum = Math.min(x, y);

        // 大きい方から小さい方を割った余を求める
        int surplus = biggerNum % smallerNum;

        // 割り切れていれば、それを返す
        if (surplus == 0) {
            return smallerNum;
        }
        // 割り切れなければ再帰的に自信を呼び出す
        surplus = getCommonDivisor(smallerNum, surplus);

        return surplus;
    }
}
```

プログラム2　Eucledean

```
public class Eratosthenes {
    public static void main(String[] args) {
        int maxP = 1000;// この値までの素数を求める
        int i, p;
        int[] a;
        a = new int[maxP];

        p = 2;
        while (p * p <= maxP) {
            // ふるいの作成

            for (i = 2; p * i < maxP; i = i + 1) {
                a[p * i] = 1;
            }
            for (p = p + 1; p * p <= maxP; p = p + 1) {
                if (a[p] == 0) {
                    break;
                }
            }
        }
        for (i = 2; i < maxP; i++) {
            if (a[i] == 0) {
                System.out.println(" " + i);
            }
        }
```

```
    }
}
```

プログラム3　Eratosthenes

【プログラムの解説】

課題 1　for文で、x,y軸へのプロットをkで指定した数のみ行い、円の中に入った時にカウントを行い、計算した。

課題 2 コマンドライン上に値を入れ、それによって値を出力出来るようにした。大きい方から小さい方を割ったあまりを求めて、割り切れ無かった場合には、再帰的に自身を呼び出すようにしてある。

課題 3

1. 2からnまでの整数を並べる。
2. 生き残っている数の中で一番小さく、まだpとして使われていないものを新たにpとおき、p以外のpの倍数を全て消す。
3. 2の操作を繰り返していき、ｐが√nを超えたら終了
4. 3の状態でふるいにかけられて残っているものが素数

プログラムはこの流れで書いた。√nを作成するのでは無く、pの2乗にする事で、同じプログラムの形にした。

【結果】

```
c83557d7bff0d329c744c8\r
プロット数=1000000000
円内数=7.85404662E8
PI=3.141618648
```

図1　MonteCarloの出力結果

```
vscode-java-debug-0.27.1\scripts\launcher.bat' 'C:\Users\acfoa\AppData\Local\Programs\A
et,server=n,suspend=y,address=localhost:54068' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\
c83557d7bff0d329c744c8\redhat.java\jdt_ws\SchoolJava_575429d3\bin' '09□lvZ2.Eucledean'
7047741 2658273
51
```

図2　Eucledeanの出力結果

```
 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 19
3 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 64
3 647 653 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821 823 827 829 839 853 857 859 863 877 881 883
887 907 911 919 929 937 941 947 953 967 971 977 983 991 997
```

図3　Eratosthenesの出力結果

【考察】

Javaで初めてコンソールを使ったプログラムを使用した。Pythonと違った点を持っていることがわかった。ふるいを思ったよりも簡単に作成できることがわかった。

また、MonteCarloでは、簡単に円周率を近似できることがわかった。

【参考文献】

1. 「アルゴリズムとデータ解析の授業スライド」
2. 「エラトステネスのふるいとその計算量」< https://mathtrain.jp/eratosthenes>
3. 「モンテカルロ法　【 Monte Carlo method 】」< http://e-words.jp/w/モンテカルロ法.html>
4. 「ユークリッドの互除法　【 Euclidean algorithm 】」< http://e-words.jp/w/ユークリッドの互除法.html>
5. 増井敏克(2020)「Pythonで始めるアルゴリズム入門」翔泳社
6.
7. 「参考文献の書き方」<http://www7a.biglobe.ne.jp/nifongo/ron/ron_04.html>