学号：2110426170　　姓名：杨博

实验 1 比特币“挖矿”算法简化模拟

比特币“挖矿”算法简化模拟

- 参数： 哈希函数采用 SHA-256 ；挖矿难度为整数 d （ >0 ），代表哈希值前导字符串为 0 的数量，d 值越小难度越小， d 值越大难度越大
- 1. 随机生成一个整数 x
- 2. 对 x 生成哈希值 h=H(x)
- 3. 对 h 的前导字符串进行判定：如为 d 个 0 则挖矿成功；否则回到步骤 1 - 输出： d=1 ， 2 ，3 ， 4 ， 5 时挖矿成功耗时（可取 3 组数平均值）

Java 实现：

整体思路是创建一个实现类 Mining，主要成员有 String num（随机整数），String hashcode（哈希值），String hashstr（去掉前导字符串 0 的哈希值）。主要方法有 String getHashCode()获得哈希值；boolean digSuccess()判断哈希值是否满足要求，满足返回 true；printResult()挖掘成功时打印整数和对应哈希值。

一、对整数 x 生成哈希值 h=H(x)

1. 生成随机整数 this.num

```
Random r= new Random();        // import java.util.Random;
String str = Integer.toString(r.nextInt(Integer.MAX_VALUE));
// MAX_VALUE 即 0xffffffff
this.num = str;
```

2. 生成哈希值 this.hashcode, this.hashstr

```
MessageDigest md = MessageDigest.getInstance("SHA-256");
// import java.security.MessageDigest;
md.update(this.num.getBytes("UTF-8"));
byte[] result = md.digest();
this.hashstr = new BigInteger(1,result).toString(16); //import java.math.BigInteger;
this.hashcode = String.format("%064x", new BigInteger(1, result));
```

二、判断得到哈希值是否满足前导字符串 0 个数等于 d

因为 hashcode.length()为 64，hashstr 是去掉前导 0 元素的哈希值，d 为前导元素 0 的个数，所以等式 hashstr+d=64 成立。有以下判定方法：

```
public boolean digSuccess(String s,int d) {
    if(s.length()+d==64)
        printResult();
    return s.length()+d==64;
}
```

三、输出符合要求的整数和哈希值

函数 digSuccess()返回 true 时输出结果，输出时左对其，使其更直观。

```
public void printResult() {
    System.out.println(String.format("随机整数: %s", this.num));
    System.out.println(String.format("  哈希值: %s", this.hashcode));
    }
```

四、开始挖矿

挖矿使用了嵌套循环，各个说明如注释所示：

```
Mining mining = new Mining();  // 实例化 mining 对象
for(int d=1;d<6;++d) {         // 挖矿难度 d=1，2，3，4，5
    long cost = 0; // 总共挖矿用时
    for(int i=0;i<loop;++i) {  // 挖矿成功次数 loop
        long time1 = System.currentTimeMillis();  //挖矿开始
        while(!flag) {         // 挖矿成功一次时停止
            String str = mining.getHashCode();     //获得哈希值
            flag = mining.digSuccess(str,d);       //判断挖矿是否成功
        }
        long time2 = System.currentTimeMillis();  //挖矿结束
        cost += time2-time1;   //每次挖矿成功一次用时
    }
}
```

五、结果

```
  难度 d: 1     成功挖矿次数: 3
随机整数: 1337177597
  哈希值: 04b2729a4f56091a05c9dce774febbf994da4e7377359aa7951ed134a5ac7e19
随机整数: 207973340
  哈希值: 07857a69b03e1c4fdbef1937ea417daa5ddc5002b952ffded38fea150e1b7f1f
随机整数: 764351681
  哈希值: 0671f3eb214e259aa9d0990666d50b4bbd515451efa33ec11e5bc7fc1ede8806
    次数: 8             平均时长:34ms

  难度 d: 2     成功挖矿次数: 3
随机整数: 1715223415
  哈希值: 007e2408c6379a2b5fc6cf65748660c2a7dadbf31c9dfd62ecf3978f13cb99d0
随机整数: 131490155
  哈希值: 002c7d25bcb19232eea4cefde6179796148eb0006c4869b526e44ef54b61ecd2
随机整数: 1959374892
  哈希值: 005f3c030334589cf58e3083bcb3c8cc7e3f18d65dcd7d4fdf8446c254b533af
    次数: 731           平均时长:50ms

  难度 d: 3     成功挖矿次数: 3
随机整数: 95166560
  哈希值: 000fc78562bee8464753c2f64350ad97aa34e16be32146c8d32b79266a7ca301
随机整数: 556121830
  哈希值: 000fd51ecb6f44ed7748ff42817ec38ab5b700b3fe4f2b1c97a8b71630cb5dbe
随机整数: 1172857403
  哈希值: 0006d8c92ff1fffe24efbb7378705bcd91890091f19c513dfe7293c21680f8de
    次数: 15119         平均时长:182ms

  难度 d: 4     成功挖矿次数: 3
随机整数: 1619572135
  哈希值: 000046312739b258f641897ff49277b0ce2855debd4edcf602787507ded7ddf5
随机整数: 1910898637
  哈希值: 0000f8621d98f1d8ca1064e80e50d4b0fc5daa987e4f5ce8e25c21e6dcfa38b2
随机整数: 1123146159
  哈希值: 0000583af4e6301b8bba1dcc5459141d117c68bdf914fb17609853527fe060b7
    次数: 336521        平均时长:1024ms

  难度 d: 5     成功挖矿次数: 3
随机整数: 892278721
  哈希值: 00000f1b77b441f5df2096e54a351d9bfdfe1a18b8318a87f93b4dfa3a96b548
随机整数: 1142780740
  哈希值: 00000370d20534c8b9d2057b4b0d44d0a69aecd67fb703a8ec5783de779e7193
随机整数: 794555959
  哈希值: 000001461211a4068afd2409aba4a8f268432f7652e76296a9111f1c99043000
    次数: 2101215               平均时长:4644ms
```

六、分析总结

1. 结果具有偶然性

循环次数，即每个难度挖矿的次数 loop=3 太小了，结果具有偶然性。例如 d=5 时，结果显示要找到一个符合要求的（哈希值前导元素为 0 等于 d）随机整数的次数高达数十万甚至上百万，对比 loop=3 来说并不是一个数量级的，每次结果误差很大。

2. 程序局限性

哈希函数生成的哈希值是随机的无规律的，因此在理论上，d=2(哈希值为：00.....)时花费的时间应该是 d=1(0.....)时的 16 倍，d=3(哈希值为 000.....)花费的时间应该是 d=2 的 16 倍。但是从实验结果可以看到 d=1,2,3 时花费时间相差不大，甚至偶尔还会出现 d=1 花费的时间比 d=2 的要长。这是因为程序运行过程中加载各种必要资源时间不同造成的时间差异，这是程序本身的局限性，无法避免。

程序

```
import java.security.MessageDigest;
import java.math.BigInteger;
import java.util.Random;

public class DigMine {
    public static void main(String[] args) throws Exception{
        Mining mining = new Mining();  // 实例化 mining 对象
        int loop = 3;   //每次难度循环次数，即符合条件的整数个数
        for(int d=3;d<4;++d) {          // 挖矿难度 d=1，2，3，4，5
            System.out.println("  难度 d: "+d+"\t 成功挖矿次数: "+loop);
            long cost = 0; // 总共挖矿用时
            int count=0;    // 总共挖矿次数
            for(int i=0;i<loop;++i) {  // 挖矿成功次数 loop
                boolean flag = false;
                long time1 = System.currentTimeMillis();  //挖矿开始
                while(!flag) {          // 挖矿成功一次时停止
                    String str = mining.getHashCode();
                    flag = mining.digSuccess(str,d);
                    count++;
                }
                long time2 = System.currentTimeMillis();  //挖矿结束
                cost += time2-time1;
            }
            System.out.println("    次数: "+count+"\t\t"+"平均时长:"+cost/loop+"ms");
            System.out.println();
        }
    }
}

class Mining{
    private String num="";
    private String hashstr="";
    private String hashcode="";

    public String getHashCode() throws Exception{
        Random r= new Random();
```

```
        String str = Integer.toString(r.nextInt(Integer.MAX_VALUE));
        this.num = str;
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(this.num.getBytes("UTF-8"));
        byte[] result = md.digest();
        this.hashstr = new BigInteger(1,result).toString(16);
        this.hashcode = String.format("%064x", new BigInteger(1, result));
        return this.hashstr;
    }

    public boolean digSuccess(String s,int d) {
        if(s.length()+d==64)
            printResult();
        return s.length()+d==64;
    }

    public void printResult() {
        System.out.println(String.format("随机整数: %s", this.num));
        System.out.println(String.format("  哈希值: %s", this.hashcode));
    }
}
```