Java大数模板

目录

[Java包 1](#_Toc464747605)

[大数 4](#_Toc464747606)

[基本函数 4](#_Toc464747607)

[基本常量 5](#_Toc464747608)

[基本操作 5](#_Toc464747609)

[Java保留小数 7](#_Toc464747610)

[Java有序map 9](#_Toc464747611)

[Java无序map 10](#_Toc464747612)

[Set 10](#_Toc464747613)

[Java queue 11](#_Toc464747614)

[Java 排序 12](#_Toc464747615)

[Java BigInteger 开方 18](#_Toc464747616)

[Java BigDecimal 开方 19](#_Toc464747617)

[Java BigDecimal pow 21](#_Toc464747618)

## Java包

**import** java.math.BigInteger;

**import java.math.BigDecimal**

**import** java.util.Scanner;

import java.util.\*;

[java.util.ArrayList](http://www.programcreek.com/java-api-examples/index.php?api=java.util.ArrayList)(6958)

[java.io.IOException](http://www.programcreek.com/java-api-examples/index.php?api=java.io.IOException)(6866)

[java.util.List](http://www.programcreek.com/java-api-examples/index.php?api=java.util.List)(6784)

[java.util.HashMap](http://www.programcreek.com/java-api-examples/index.php?api=java.util.HashMap)(5590)

[java.util.Map](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Map)(5413)

[java.io.File](http://www.programcreek.com/java-api-examples/index.php?api=java.io.File)(5097)

[java.io.InputStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.InputStream)(4234)

[java.util.Set](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Set)(3915)

[java.util.Arrays](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Arrays)(3884)

[java.util.Iterator](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Iterator)(3856)

[java.util.Collections](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Collections)(3643)

[java.util.Date](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Date)(3461)

[java.util.HashSet](http://www.programcreek.com/java-api-examples/index.php?api=java.util.HashSet)(3278)

[java.io.BufferedReader](http://www.programcreek.com/java-api-examples/index.php?api=java.io.BufferedReader)(3257)

[java.util.Collection](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Collection)(3192)

[java.net.URL](http://www.programcreek.com/java-api-examples/index.php?api=java.net.URL)(3168)

[java.io.FileInputStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.FileInputStream)(3044)

[java.io.InputStreamReader](http://www.programcreek.com/java-api-examples/index.php?api=java.io.InputStreamReader)(3023)

[org.junit.Test](http://www.programcreek.com/java-api-examples/index.php?api=org.junit.Test)(3008)

[java.io.FileOutputStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.FileOutputStream)(2843)

[java.io.FileNotFoundException](http://www.programcreek.com/java-api-examples/index.php?api=java.io.FileNotFoundException)(2669)

[java.io.OutputStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.OutputStream)(2563)

[java.util.regex.Pattern](http://www.programcreek.com/java-api-examples/index.php?api=java.util.regex.Pattern)(2469)

[java.io.Serializable](http://www.programcreek.com/java-api-examples/index.php?api=java.io.Serializable)(2437)

[java.util.LinkedList](http://www.programcreek.com/java-api-examples/index.php?api=java.util.LinkedList)(2372)

[java.text.SimpleDateFormat](http://www.programcreek.com/java-api-examples/index.php?api=java.text.SimpleDateFormat)(2245)

[java.util.Properties](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Properties)(2190)

[java.util.Random](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Random)(2171)

[java.lang.reflect.Method](http://www.programcreek.com/java-api-examples/index.php?api=java.lang.reflect.Method)(2141)

[java.io.ByteArrayOutputStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.ByteArrayOutputStream)(2112)

[java.util.regex.Matcher](http://www.programcreek.com/java-api-examples/index.php?api=java.util.regex.Matcher)(2012)

[android.os.Bundle](http://www.programcreek.com/java-api-examples/index.php?api=android.os.Bundle)(2007)

[java.util.logging.Logger](http://www.programcreek.com/java-api-examples/index.php?api=java.util.logging.Logger)(1999)

[java.io.UnsupportedEncodingException](http://www.programcreek.com/java-api-examples/index.php?api=java.io.UnsupportedEncodingException)(1968)

[org.junit.Before](http://www.programcreek.com/java-api-examples/index.php?api=org.junit.Before)(1920)

[java.util.Comparator](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Comparator)(1896)

[java.io.ByteArrayInputStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.ByteArrayInputStream)(1868)

[java.io.PrintWriter](http://www.programcreek.com/java-api-examples/index.php?api=java.io.PrintWriter)(1862)

[java.util.Calendar](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Calendar)(1854)

[android.app.Activity](http://www.programcreek.com/java-api-examples/index.php?api=android.app.Activity)(1843)

[java.net.MalformedURLException](http://www.programcreek.com/java-api-examples/index.php?api=java.net.MalformedURLException)(1828)

[android.content.Context](http://www.programcreek.com/java-api-examples/index.php?api=android.content.Context)(1780)

[android.view.View](http://www.programcreek.com/java-api-examples/index.php?api=android.view.View)(1731)

[java.util.Locale](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Locale)(1719)

[java.util.Enumeration](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Enumeration)(1709)

[java.util.Map.Entry](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Map.Entry)(1705)

[java.io.FileWriter](http://www.programcreek.com/java-api-examples/index.php?api=java.io.FileWriter)(1677)

[java.io.FileReader](http://www.programcreek.com/java-api-examples/index.php?api=java.io.FileReader)(1651)

[android.util.Log](http://www.programcreek.com/java-api-examples/index.php?api=android.util.Log)(1614)

[android.content.Intent](http://www.programcreek.com/java-api-examples/index.php?api=android.content.Intent)(1601)

[java.lang.reflect.InvocationTargetException](http://www.programcreek.com/java-api-examples/index.php?api=java.lang.reflect.InvocationTargetException)(1594)

[java.util.logging.Level](http://www.programcreek.com/java-api-examples/index.php?api=java.util.logging.Level)(1557)

[java.lang.reflect.Field](http://www.programcreek.com/java-api-examples/index.php?api=java.lang.reflect.Field)(1499)

[java.io.StringWriter](http://www.programcreek.com/java-api-examples/index.php?api=java.io.StringWriter)(1499)

[android.widget.TextView](http://www.programcreek.com/java-api-examples/index.php?api=android.widget.TextView)(1442)

[java.util.LinkedHashMap](http://www.programcreek.com/java-api-examples/index.php?api=java.util.LinkedHashMap)(1409)

[java.io.Reader](http://www.programcreek.com/java-api-examples/index.php?api=java.io.Reader)(1390)

[java.net.URI](http://www.programcreek.com/java-api-examples/index.php?api=java.net.URI)(1377)

[java.io.Writer](http://www.programcreek.com/java-api-examples/index.php?api=java.io.Writer)(1339)

[java.text.ParseException](http://www.programcreek.com/java-api-examples/index.php?api=java.text.ParseException)(1318)

[junit.framework.TestCase](http://www.programcreek.com/java-api-examples/index.php?api=junit.framework.TestCase)(1318)

[java.io.OutputStreamWriter](http://www.programcreek.com/java-api-examples/index.php?api=java.io.OutputStreamWriter)(1295)

[java.io.StringReader](http://www.programcreek.com/java-api-examples/index.php?api=java.io.StringReader)(1279)

[java.io.BufferedWriter](http://www.programcreek.com/java-api-examples/index.php?api=java.io.BufferedWriter)(1265)

[java.util.Vector](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Vector)(1254)

[java.util.StringTokenizer](http://www.programcreek.com/java-api-examples/index.php?api=java.util.StringTokenizer)(1251)

[java.text.DateFormat](http://www.programcreek.com/java-api-examples/index.php?api=java.text.DateFormat)(1246)

[java.util.concurrent.TimeUnit](http://www.programcreek.com/java-api-examples/index.php?api=java.util.concurrent.TimeUnit)(1237)

[java.io.BufferedInputStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.BufferedInputStream)(1235)

[java.util.TreeMap](http://www.programcreek.com/java-api-examples/index.php?api=java.util.TreeMap)(1227)

[org.xml.sax.SAXException](http://www.programcreek.com/java-api-examples/index.php?api=org.xml.sax.SAXException)(1218)

[javax.servlet.http.HttpServletRequest](http://www.programcreek.com/java-api-examples/index.php?api=javax.servlet.http.HttpServletRequest)(1175)

[java.io.PrintStream](http://www.programcreek.com/java-api-examples/index.php?api=java.io.PrintStream)(1168)

[java.util.TreeSet](http://www.programcreek.com/java-api-examples/index.php?api=java.util.TreeSet)(1160)

[android.widget.Toast](http://www.programcreek.com/java-api-examples/index.php?api=android.widget.Toast)(1157)

[java.util.Hashtable](http://www.programcreek.com/java-api-examples/index.php?api=java.util.Hashtable)(1154)

[java.lang.reflect.Constructor](http://www.programcreek.com/java-api-examples/index.php?api=java.lang.reflect.Constructor)(1139)

[java.net.URLEncoder](http://www.programcreek.com/java-api-examples/index.php?api=java.net.URLEncoder)(1134)

[java.security.NoSuchAlgorithmException](http://www.programcreek.com/java-api-examples/index.php?api=java.security.NoSuchAlgorithmException)(1134)

[org.w3c.dom.Document](http://www.programcreek.com/java-api-examples/index.php?api=org.w3c.dom.Document)(1130)

[android.widget.Button](http://www.programcreek.com/java-api-examples/index.php?api=android.widget.Button)(1129)

[org.junit.After](http://www.programcreek.com/java-api-examples/index.php?api=org.junit.After)(1128)

[javax.servlet.http.HttpServletResponse](http://www.programcreek.com/java-api-examples/index.php?api=javax.servlet.http.HttpServletResponse)(1109)

[java.awt.Color](http://www.programcreek.com/java-api-examples/index.php?api=java.awt.Color)(1099)

[java.net.URISyntaxException](http://www.programcreek.com/java-api-examples/index.php?api=java.net.URISyntaxException)(1085)

[javax.servlet.ServletException](http://www.programcreek.com/java-api-examples/index.php?api=javax.servlet.ServletException)(1081)

[javax.xml.parsers.DocumentBuilderFactory](http://www.programcreek.com/java-api-examples/index.php?api=javax.xml.parsers.DocumentBuilderFactory)(1076)

[java.lang.annotation.Retention](http://www.programcreek.com/java-api-examples/index.php?api=java.lang.annotation.Retention)(1075)

[java.security.MessageDigest](http://www.programcreek.com/java-api-examples/index.php?api=java.security.MessageDigest)(1072)

[java.util.concurrent.Executors](http://www.programcreek.com/java-api-examples/index.php?api=java.util.concurrent.Executors)(1062)

[java.net.UnknownHostException](http://www.programcreek.com/java-api-examples/index.php?api=java.net.UnknownHostException)(1057)

[org.slf4j.Logger](http://www.programcreek.com/java-api-examples/index.php?api=org.slf4j.Logger)(1054)

[java.sql.SQLException](http://www.programcreek.com/java-api-examples/index.php?api=java.sql.SQLException)(1043)

[org.slf4j.LoggerFactory](http://www.programcreek.com/java-api-examples/index.php?api=org.slf4j.LoggerFactory)(1042)

[java.util.UUID](http://www.programcreek.com/java-api-examples/index.php?api=java.util.UUID)(1040)

[java.net.InetAddress](http://www.programcreek.com/java-api-examples/index.php?api=java.net.InetAddress)(1026)

[junit.framework.Assert](http://www.programcreek.com/java-api-examples/index.php?api=junit.framework.Assert)(1011)

[android.view.ViewGroup](http://www.programcreek.com/java-api-examples/index.php?api=android.view.ViewGroup)(1005)

[java.util.concurrent.ConcurrentHashMap](http://www.programcreek.com/java-api-examples/index.php?api=java.util.concurrent.ConcurrentHashMap)(1001)

[java.awt.event.ActionEvent](http://www.programcreek.com/java-api-examples/index.php?api=java.awt.event.ActionEvent)(995)

## 大数

在用C或者C++处理大数时感觉非常麻烦，但是在JAVA中有两个类BigInteger和BigDecimal分别表示大整数类和大浮点数类，至于两个类的对象能表示最大范围不清楚，理论上能够表示无线大的数，只要计算机内存足够大。这两个类都在java.math.\*包中，因此每次必须在开头处引用该包。

### 基本函数

1.valueOf(parament); 将参数转换为制定的类型

  比如 int a=3;

  BigInteger b=BigInteger.valueOf(a);

  则b=3;

  String s=”12345”;

  BigInteger c=BigInteger.valueOf(s);

  则c=12345；

2.add(); 大整数相加

BigInteger a=new BigInteger(“23”);

BigInteger b=new BigInteger(“34”);

a. add(b);

3.subtract(); 相减

4.multiply(); 相乘

5.divide();    相除取整

6.remainder(); 取余

7.pow();   a.pow(b)=a^b

8.gcd();   最大公约数

9.abs(); 绝对值

10.negate(); 取反数

11.mod(); a.mod(b)=a%b=a.remainder(b);

12.max(); min();

13.public int comareTo();

14.boolean equals(); 是否相等

15.BigInteger构造函数：

一般用到以下两种：

BigInteger(String val);

将指定字符串转换为十进制表示形式；

BigInteger(String val,int radix);

将指定基数的 BigInteger 的字符串表示形式转换为 BigInteger

### 基本常量

A=BigInteger.ONE    1

B=BigInteger.TEN    10

C=BigInteger.ZERO   0

### 基本操作

1.   读入：

用Scanner类定义对象进行控制台读入,Scanner类在java.util.\*包中

Scanner cin=new Scanner(System.in);// 读入

while(cin.hasNext())   //等同于!=EOF

{

  int n;

 BigInteger m;

  n=cin.nextInt(); //读入一个int;

  m=cin.BigInteger();//读入一个BigInteger;

  System.out.print(m.toString());

}

Ⅳ.运用

四则预算：

**import** java.math.BigDecimal;

**import** java.math.BigInteger;

**import** java.text.DecimalFormat;

**import** java.util.Scanner;

**public** **class** Main5 {

**public** **static** **void** main(String args[]) {

Scanner cin = **new** Scanner(System.***in***);

BigInteger a, b;

**int** c;

**char** op;

String s;

**while** (cin.hasNext()) {

a = cin.nextBigInteger();

s = cin.next();

op = s.charAt(0);

**if** (op == '+') {

b = cin.nextBigInteger();

System.***out***.println(a.add(b));

} **else** **if** (op == '-') {

b = cin.nextBigInteger();

System.***out***.println(a.subtract(b));

} **else** **if** (op == '\*') {

b = cin.nextBigInteger();

System.***out***.println(a.multiply(b));

} **else** {

BigDecimal a1, b1, eps;

String s1, s2, temp;

s1 = a.toString();

a1 = **new** BigDecimal(s1);

b = cin.nextBigInteger();

s2 = b.toString();

b1 = **new** BigDecimal(s2);

c = cin.nextInt();

eps = a1.divide(b1, c, 4);

// System.out.println(a + " " + b + " " + c);

// System.out.println(a1.doubleValue() + " " + b1.doubleValue()

// + " " + c);

System.***out***.print(a.divide(b) + " " + a.mod(b) + " ");

**if** (c != 0) {

temp = "0.";

**for** (**int** i = 0; i < c; i++)

temp += "0";

DecimalFormat gd = **new** DecimalFormat(temp);

System.***out***.println(gd.format(eps));

} **else**

System.***out***.println(eps);

}

}

}

}

//==========================================================================

//PKU1311八进制浮点数化为十进制浮点数，高精度  
**import** java.math.BigDecimal;

**import** java.util.Scanner;

**public** **class** Main5 {

**public** **static** **void** main(String[] args) {

Scanner cin = **new** Scanner(System.***in***);

BigDecimal temp, sum, ans, num; // java大数

String str;

**int** i, len;

**while** (cin.hasNext()) {

str = cin.next();

len = str.length();

temp = BigDecimal.*valueOf*(8.0);

sum = BigDecimal.***ONE***;

ans = BigDecimal.***ZERO***;

**for** (i = 2; i < len; i++) {

**int** val = str.charAt(i) - '0';

num = BigDecimal.*valueOf*(val);

sum = sum.multiply(temp); // 8的n次幂

ans = ans.add(num.divide(sum)); // 按权累加

}

System.***out***.printf("%s [8] = ", str);

System.***out***.println(ans + " [10]");

}

}

}

## Java保留小数

**public** **class** PreciseCompute {

//默认除法运算精度

**private** **static** **final** **int** ***DEF\_DIV\_SCALE*** = 10;

/\*\*

\* 提供精确的加法运算。

\* **@param** v1 被加数

\* **@param** v2 加数

\* **@return** 两个参数的和

\*/

**public** **static** **double** add(**double** v1, **double** v2) {

BigDecimal b1 = **new** BigDecimal(Double.*toString*(v1));

BigDecimal b2 = **new** BigDecimal(Double.*toString*(v2));

**return** b1.add(b2).doubleValue();

}

/\*\*

\* 提供精确的减法运算。

\* **@param** v1 被减数

\* **@param** v2 减数

\* **@return** 两个参数的差

\*/

**public** **static** **double** sub(**double** v1, **double** v2) {

BigDecimal b1 = **new** BigDecimal(Double.*toString*(v1));

BigDecimal b2 = **new** BigDecimal(Double.*toString*(v2));

**return** b1.subtract(b2).doubleValue();

}

/\*\*

\* 提供精确的乘法运算。

\* **@param** v1 被乘数

\* **@param** v2 乘数

\* **@return** 两个参数的积

\*/

**public** **static** **double** mul(**double** v1, **double** v2) {

BigDecimal b1 = **new** BigDecimal(Double.*toString*(v1));

BigDecimal b2 = **new** BigDecimal(Double.*toString*(v2));

**return** b1.multiply(b2).doubleValue();

}

/\*\*

\* 提供（相对）精确的除法运算，当发生除不尽的情况时，精确到

\* 小数点以后10位，以后的数字四舍五入。

\* **@param** v1 被除数

\* **@param** v2 除数

\* **@return** 两个参数的商

\*/

**public** **static** **double** div(**double** v1, **double** v2) {

**return** *div*(v1, v2, ***DEF\_DIV\_SCALE***);

}

/\*\*

\* 提供（相对）精确的除法运算。当发生除不尽的情况时，由scale参数指

\* 定精度，以后的数字四舍五入。

\* **@param** v1 被除数

\* **@param** v2 除数

\* **@param** scale 表示表示需要精确到小数点以后几位。

\* **@return** 两个参数的商

\*/

**public** **static** **double** div(**double** v1, **double** v2, **int** scale) {

**if** (scale < 0) {

**throw** **new** IllegalArgumentException(

"The scale must be a positive integer or zero");

}

BigDecimal b1 = **new** BigDecimal(Double.*toString*(v1));

BigDecimal b2 = **new** BigDecimal(Double.*toString*(v2));

**return** b1.divide(b2, scale, BigDecimal.ROUND\_HALF\_UP).doubleValue();

}

/\*\*

\* 提供精确的小数位四舍五入处理。

\* **@param** v 需要四舍五入的数字

\* **@param** scale 小数点后保留几位

\* **@return** 四舍五入后的结果

\*/

**public** **static** **double** round(**double** v, **int** scale) {

**if** (scale < 0) {

**throw** **new** IllegalArgumentException(

"The scale must be a positive integer or zero");

}

BigDecimal b = **new** BigDecimal(Double.*toString*(v));

BigDecimal ne = **new** BigDecimal("1");

**return** b.divide(one, scale, BigDecimal.ROUND\_HALF\_UP).doubleValue();

}

}

## Java有序map

public class TreeMapTest {  
    public static void main(String[] args) {  
        Map<String, String> map = new TreeMap<String, String>(  
                new Comparator<String>() {  
                    public int compare(String obj1, String obj2) {  
                        // 降序排序  
                        return obj2.compareTo(obj1);  
                    }  
                });  
        map.put("c", "ccccc");  
        map.put("a", "aaaaa");  
        map.put("b", "bbbbb");  
        map.put("d", "ddddd");  
  
        Set<String> keySet = map.keySet();  
        Iterator<String> iter = keySet.iterator();  
        while (iter.hasNext()) {  
            String key = iter.next();  
            System.out.println(key + ":" + map.get(key));  
        }  
    }  
}

## Java无序map

public class HashMapTest {  
    public static void main(String[] args) {  
        Map<String, String> map = new HashMap<String, String>();  
        map.put("c", "ccccc");  
        map.put("a", "aaaaa");  
        map.put("b", "bbbbb");  
        map.put("d", "ddddd");  
       List<Map.Entry<String,String>> list = new ArrayList<Map.Entry<String,String>>(map.entrySet());  
        Collections.sort(list,new Comparator<Map.Entry<String,String>>() {  
            //升序排序  
            public int compare(Entry<String, String> o1,  
                    Entry<String, String> o2) {  
                return o1.getValue().compareTo(o2.getValue());  
            }  
        });  
        for(Map.Entry<String,String> mapping:list){   
               System.out.println(mapping.getKey()+":"+mapping.getValue());  
          }   
     }  
}

## Set

HashSet:特点：无序的，长度可变的，不可重复的

1. // 创建一个容器对象
2. java.util.HashSet<String> sets = **new** java.util.HashSet<String>();
4. // 装入10个元素
5. **for** (**int** i = 0; i < 10; i++) {
6. String s = "元素a" + i;
7. sets.add(s);
8. }
10. //检验若输入加入重复的元素，在集合中的是原来的还是新加入的（结果表明是原来的）
11. **boolean** b = sets.add("新来的");
12. **boolean** b1 = sets.add("新来的");
13. System.out.println(b + "<>" + b1);
14. // 遍历1
15. // 得到迭代器对象
16. java.util.Iterator<String> iter = sets.iterator();
17. **while** (iter.hasNext()) {
18. String str = iter.next();
19. System.out.print(str + "\t");
20. }
22. System.out.println();
24. //遍历2
25. **for**(String str:sets){
27. System.out.print(str+"\t");
28. }

## Java queue

import java.util.LinkedList;

import java.util.Queue;

public class QueueTest {

public static void main(String[] args) {

//add()和remove()方法在失败的时候会抛出异常(不推荐)

Queue<String> queue = new LinkedList<String>();

//添加元素

queue.offer("a");

queue.offer("b");

queue.offer("c");

queue.offer("d");

queue.offer("e");

for(String q : queue){

System.out.println(q);

}

System.out.println("===");

System.out.println("poll="+queue.poll()); //返回第一个元素，并在队列中删除

for(String q : queue){

System.out.println(q);

}

System.out.println("===");

System.out.println("element="+queue.element()); //返回第一个元素

for(String q : queue){

System.out.println(q);

}

System.out.println("===");

System.out.println("peek="+queue.peek()); //返回第一个元素

for(String q : queue){

System.out.println(q);

}

}

}

## Java 排序

/\*\*    
 \* 冒泡法排序<br/>    
 \* <li>比较相邻的元素。如果第一个比第二个大，就交换他们两个。</li>    
 \* <li>对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。</li>    
 \* <li>针对所有的元素重复以上的步骤，除了最后一个。</li>    
 \* <li>持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。</li>    
 \*     
 \* @param numbers    
 \*            需要排序的整型数组    
 \*/    
public static void bubbleSort(int[] numbers) {     
    int temp; // 记录临时中间值     
    int size = numbers.length; // 数组大小     
    for (int i = 0; i < size - 1; i++) {     
        for (int j = i + 1; j < size; j++) {     
            if (numbers[i] < numbers[j]) { // 交换两数的位置     
                temp = numbers[i];     
                numbers[i] = numbers[j];     
                numbers[j] = temp;     
            }     
        }     
    }     
}    
快速排序使用分治法策略来把一个序列分为两个子序列。

/\*\*    
 \* 快速排序<br/>    
 \* <ul>    
 \* <li>从数列中挑出一个元素，称为“基准”</li>    
 \* <li>重新排序数列，所有元素比基准值小的摆放在基准前面，所有元素比基准值大的摆在基准的后面（相同的数可以到任一边）。在这个分割之后，    
 \* 该基准是它的最后位置。这个称为分割（partition）操作。</li>    
 \* <li>递归地把小于基准值元素的子数列和大于基准值元素的子数列排序。</li>    
 \* </ul>    
 \*     
 \* @param numbers    
 \* @param start    
 \* @param end    
 \*/    
public static void quickSort(int[] numbers, int start, int end) {     
    if (start < end) {     
        int base = numbers[start]; // 选定的基准值（第一个数值作为基准值）     
        int temp; // 记录临时中间值     
        int i = start, j = end;     
        do {     
            while ((numbers[i] < base) && (i < end))     
                i++;     
            while ((numbers[j] > base) && (j > start))     
                j--;     
            if (i <= j) {     
                temp = numbers[i];     
                numbers[i] = numbers[j];     
                numbers[j] = temp;     
                i++;     
                j--;     
            }     
        } while (i <= j);     
        if (start < j)     
            quickSort(numbers, start, j);     
        if (end > i)     
            quickSort(numbers, i, end);     
    }     
}    
选择排序是一种简单直观的排序方法，每次寻找序列中的最小值，然后放在最末尾的位置。

/\*\*    
 \* 选择排序<br/>    
 \* <li>在未排序序列中找到最小元素，存放到排序序列的起始位置</li>    
 \* <li>再从剩余未排序元素中继续寻找最小元素，然后放到排序序列末尾。</li>    
 \* <li>以此类推，直到所有元素均排序完毕。</li>    
  
 \*     
 \* @param numbers    
 \*/    
public static void selectSort(int[] numbers) {     
    int size = numbers.length, temp;     
    for (int i = 0; i < size; i++) {     
        int k = i;     
        for (int j = size - 1; j >i; j--)  {     
            if (numbers[j] < numbers[k])  k = j;     
        }     
        temp = numbers[i];     
        numbers[i] = numbers[k];     
        numbers[k] = temp;     
    }     
}    
插入排序的工作原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。其具体步骤参见代码及注释。

/\*\*    
 \* 插入排序<br/>    
 \* <ul>    
 \* <li>从第一个元素开始，该元素可以认为已经被排序</li>    
 \* <li>取出下一个元素，在已经排序的元素序列中从后向前扫描</li>    
 \* <li>如果该元素（已排序）大于新元素，将该元素移到下一位置</li>    
 \* <li>重复步骤3，直到找到已排序的元素小于或者等于新元素的位置</li>    
 \* <li>将新元素插入到该位置中</li>    
 \* <li>重复步骤2</li>    
 \* </ul>    
 \*     
 \* @param numbers    
 \*/    
public static void insertSort(int[] numbers) {     
    int size = numbers.length, temp, j;     
    for(int i=1; i<size; i++) {     
        temp = numbers[i];     
        for(j = i; j > 0 && temp < numbers[j-1]; j--)     
            numbers[j] = numbers[j-1];     
        numbers[j] = temp;     
    }     
}

归并排序是建立在归并操作上的一种有效的排序算法，归并是指将两个已经排序的序列合并成一个序列的操作。参考代码如下：

/\*\*    
 \* 归并排序<br/>    
 \* <ul>    
 \* <li>申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列</li>    
 \* <li>设定两个指针，最初位置分别为两个已经排序序列的起始位置</li>    
 \* <li>比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置</li>    
 \* <li>重复步骤3直到某一指针达到序列尾</li>    
 \* <li>将另一序列剩下的所有元素直接复制到合并序列尾</li>    
 \* </ul>    
 \*     
 \* @param numbers    
 \*/    
public static void mergeSort(int[] numbers, int left, int right) {     
    int t = 1;// 每组元素个数     
    int size = right - left + 1;     
    while (t < size) {     
        int s = t;// 本次循环每组元素个数     
        t = 2 \* s;     
        int i = left;     
        while (i + (t - 1) < size) {     
            merge(numbers, i, i + (s - 1), i + (t - 1));     
            i += t;     
        }     
        if (i + (s - 1) < right)     
            merge(numbers, i, i + (s - 1), right);     
    }     
}     
/\*\*    
 \* 归并算法实现    
 \*     
 \* @param data    
 \* @param p    
 \* @param q    
 \* @param r    
 \*/    
private static void merge(int[] data, int p, int q, int r) {     
    int[] B = new int[data.length];     
    int s = p;     
    int t = q + 1;     
    int k = p;     
    while (s <= q && t <= r) {     
        if (data[s] <= data[t]) {     
            B[k] = data[s];     
            s++;     
        } else {     
            B[k] = data[t];     
            t++;     
        }     
        k++;     
    }     
    if (s == q + 1)     
        B[k++] = data[t++];     
    else    
        B[k++] = data[s++];     
    for (int i = p; i <= r; i++)     
        data[i] = B[i];     
}    
 将之前介绍的所有排序算法整理成NumberSort类，代码

package test.sort;     
import java.util.Random;     
//Java实现的排序类    
public class NumberSort {     
    //私有构造方法，禁止实例化    
    private NumberSort() {     
        super();     
    }      
    //冒泡法排序   
    public static void bubbleSort(int[] numbers) {     
        int temp; // 记录临时中间值     
        int size = numbers.length; // 数组大小     
        for (int i = 0; i < size - 1; i++) {     
            for (int j = i + 1; j < size; j++) {     
                if (numbers[i] < numbers[j]) { // 交换两数的位置     
                    temp = numbers[i];     
                    numbers[i] = numbers[j];     
                    numbers[j] = temp;     
                }     
            }     
        }     
    }     
    //快速排序  
    public static void quickSort(int[] numbers, int start, int end) {     
        if (start < end) {     
            int base = numbers[start]; // 选定的基准值（第一个数值作为基准值）     
            int temp; // 记录临时中间值     
            int i = start, j = end;     
            do {     
                while ((numbers[i] < base) && (i < end))     
                    i++;     
                while ((numbers[j] > base) && (j > start))     
                    j--;     
                if (i <= j) {     
                    temp = numbers[i];     
                    numbers[i] = numbers[j];     
                    numbers[j] = temp;     
                    i++;     
                    j--;     
                }     
            } while (i <= j);     
            if (start < j)     
                quickSort(numbers, start, j);     
            if (end > i)     
                quickSort(numbers, i, end);     
        }     
    }     
    //选择排序   
    public static void selectSort(int[] numbers) {     
        int size = numbers.length, temp;     
        for (int i = 0; i < size; i++) {     
            int k = i;     
            for (int j = size - 1; j > i; j--) {     
                if (numbers[j] < numbers[k])     
                    k = j;     
            }     
            temp = numbers[i];     
            numbers[i] = numbers[k];     
            numbers[k] = temp;     
        }     
    }     
    //插入排序      
    // @param numbers    
    public static void insertSort(int[] numbers) {     
        int size = numbers.length, temp, j;     
        for (int i = 1; i < size; i++) {     
            temp = numbers[i];     
            for (j = i; j > 0 && temp < numbers[j - 1]; j--)     
                numbers[j] = numbers[j - 1];     
            numbers[j] = temp;     
        }     
    }     
    //归并排序    
    public static void mergeSort(int[] numbers, int left, int right) {     
        int t = 1;// 每组元素个数     
        int size = right - left + 1;     
        while (t < size) {     
            int s = t;// 本次循环每组元素个数     
            t = 2 \* s;     
            int i = left;     
            while (i + (t - 1) < size) {     
                merge(numbers, i, i + (s - 1), i + (t - 1));     
                i += t;     
            }     
            if (i + (s - 1) < right)     
                merge(numbers, i, i + (s - 1), right);     
        }     
    }      
    //归并算法实现    
    private static void merge(int[] data, int p, int q, int r) {     
        int[] B = new int[data.length];     
        int s = p;     
        int t = q + 1;     
        int k = p;     
        while (s <= q && t <= r) {     
            if (data[s] <= data[t]) {     
                B[k] = data[s];     
                s++;     
            } else {     
                B[k] = data[t];     
                t++;     
            }     
            k++;     
        }     
        if (s == q + 1)     
            B[k++] = data[t++];     
        else    
            B[k++] = data[s++];     
        for (int i = p; i <= r; i++)     
            data[i] = B[i];     
    }    
}

## Java BigInteger 开方

合被开方数不超过1000位的。

**import** java.util.\*;

**import** java.math.\*;

**public** **class** Kf {

**public** **static** **void** main(String[] args) {

Scanner cin = **new** Scanner(System.***in***);

**int** t = cin.nextInt();

BigInteger remain = BigInteger.***ZERO***;

BigInteger odd = BigInteger.***ZERO***;

BigInteger ans = BigInteger.***ZERO***;

**for** (**int** i = 0; i < t; i++) {

String s = cin.next();

remain = BigInteger.***ZERO***;

odd = BigInteger.***ZERO***;

ans = BigInteger.***ZERO***;

**int** group = 0, k = 0;

**if** (s.length() % 2 == 1) {

group = s.charAt(0) - '0';

k = -1;

} **else** {

group = (s.charAt(0) - '0') \* 10 + s.charAt(1) - '0';

k = 0;

}

**for** (**int** j = 0; j < (s.length() + 1) / 2; j++) {

**if** (j != 0)

group = (s.charAt(j \* 2 + k) - '0') \* 10 + s.charAt(j \* 2 + k + 1) - '0';

odd = BigInteger.*valueOf*(20).multiply(ans).add(BigInteger.***ONE***);

remain = BigInteger.*valueOf*(100).multiply(remain).add(BigInteger.*valueOf*(group));

**int** count = 0;

**while** (remain.compareTo(odd) >= 0) {

count++;

remain = remain.subtract(odd);

odd = odd.add(BigInteger.*valueOf*(2));

}

ans = ans.multiply(BigInteger.***TEN***).add(BigInteger.*valueOf*(count));

}

System.***out***.println(ans);

**if** (i != t - 1)

System.***out***.println();

}

}

}

## Java BigDecimal 开方

import java.math.BigDecimal;

import java.math.BigInteger;

public class BigSquareRoot {

final static BigInteger HUNDRED = BigInteger.valueOf(100);

public static BigDecimal sqrt(BigDecimal number, int scale, int roundingMode) {

if (number.compareTo(BigDecimal.ZERO) < 0)

throw new ArithmeticException("sqrt with negative");

BigInteger integer = number.toBigInteger();

StringBuffer sb = new StringBuffer();

String strInt = integer.toString();

int lenInt = strInt.length();

if (lenInt % 2 != 0) {

strInt = '0' + strInt;

lenInt++;

}

BigInteger res = BigInteger.ZERO;

BigInteger rem = BigInteger.ZERO;

for (int i = 0; i < lenInt / 2; i++) {

res = res.multiply(BigInteger.TEN);

rem = rem.multiply(HUNDRED);

BigInteger temp = new BigInteger(strInt.substring(i \* 2, i \* 2 + 2));

rem = rem.add(temp);

BigInteger j = BigInteger.TEN;

while (j.compareTo(BigInteger.ZERO) > 0) {

j = j.subtract(BigInteger.ONE);

if (((res.add(j)).multiply(j)).compareTo(rem) <= 0) {

break;

}

}

res = res.add(j);

rem = rem.subtract(res.multiply(j));

res = res.add(j);

sb.append(j);

}

sb.append('.');

BigDecimal fraction = number.subtract(number.setScale(0, BigDecimal.ROUND\_DOWN));

int fracLen = (fraction.scale() + 1) / 2;

fraction = fraction.movePointRight(fracLen \* 2);

String strFrac = fraction.toPlainString();

for (int i = 0; i <= scale; i++) {

res = res.multiply(BigInteger.TEN);

rem = rem.multiply(HUNDRED);

if (i < fracLen) {

BigInteger temp = new BigInteger(strFrac.substring(i \* 2, i \* 2 + 2));

rem = rem.add(temp);

}

BigInteger j = BigInteger.TEN;

while (j.compareTo(BigInteger.ZERO) > 0) {

j = j.subtract(BigInteger.ONE);

if (((res.add(j)).multiply(j)).compareTo(rem) <= 0) {

break;

}

}

res = res.add(j);

rem = rem.subtract(res.multiply(j));

res = res.add(j);

sb.append(j);

}

return new BigDecimal(sb.toString()).setScale(scale, roundingMode);

}

public static BigDecimal sqrt(BigDecimal number, int scale) {

return sqrt(number, scale, BigDecimal.ROUND\_HALF\_UP);

}

public static BigDecimal sqrt(BigDecimal number) {

int scale = number.scale() \* 2;

if (scale < 50)

scale = 50;

return sqrt(number, scale, BigDecimal.ROUND\_HALF\_UP);

}

public static void main(String args[]) {

// BigDecimal num = new BigDecimal("6510354513.6564897413514568413");

BigDecimal num = new BigDecimal("5");

long time = System.nanoTime();

BigDecimal root = sqrt(num, 1000);

time = System.nanoTime() - time;

System.out.println(root);

// System.out.println(root.pow(2));

System.out.println(time);

}

}

## Java BigDecimal pow

import java.math.\*;

public class BigDecimalDemo {

public static void main(String[] args) {

// create 2 BigDecimal Objects

BigDecimal bg1, bg2;

MathContext mc = new MathContext(4); // 4 precision

bg1 = new BigDecimal("2.17");

// apply pow method on bg1 using mc

bg2 = bg1.pow(3, mc);

String str = "The value of " + bg1 + " to the power of 3, rounded

to " + bg2;

// print bg2 value

System.out.println( str );

}

}