03讲迭代法：不用编程语言的自带函数，你会如何计算平方根

你好，我是黄申。

今天我们来说一个和编程结合得非常紧密的数学概念。在解释这个重要的概念之前，我们先来看个有趣的小故事。

> 古印度国王舍罕酷爱下棋，他打算重赏国际象棋的发明人宰相西萨·班·达依尔。这位聪明的大臣指着象棋盘对国王说："陛下，我不要别的赏赐，请您在这张棋盘的第一个小格内放入一粒麦子，在第二个小格内放入两粒，第三小格内放入给四粒，以此类推，每一小格内都比前一小格加一倍的麦子，直至放满64个格子，然后将棋盘上所有的麦粒都赏给您的仆人我吧！"

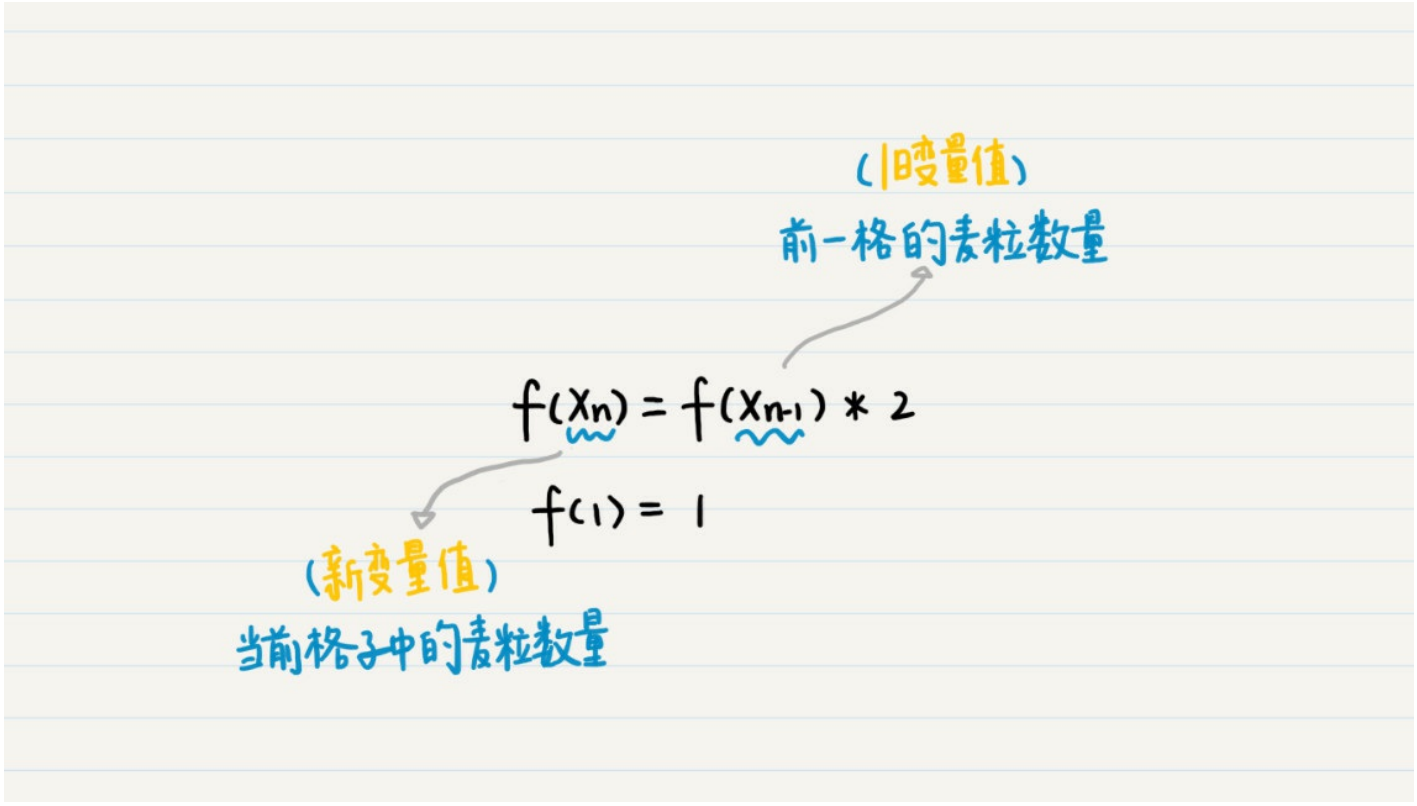
国王自以为小事一桩，痛快地答应了。可是，当开始放麦粒之后，国王发现，还没放到第二十格，一袋麦子已经空了。随着，一袋又一袋的麦子被放入棋盘的格子里，国王很快看出来，即便拿来全印度的粮食，也兑现不了对达依尔的诺言。

放满这64格到底需要多少粒麦子呢？这是个相当相当大的数字，想要手动算出结果并不容易。如果你觉得自己厉害，可以试着拿笔算算。其实，这整个算麦粒的过程，在数学上，是有对应方法的，这也正是我们今天要讲的概念：**迭代法**（Iterative Method）。

到底什么是迭代法？

迭代法，简单来说，其实就是不断地用旧的变量值，递推计算新的变量值。

我这么说可能还是比较抽象，不容易理解。我们还回到刚才的故事。大臣要求每一格的麦子都是前一格的两倍，那么前一格里麦子的数量就是旧的变量值，我们可以先记作X_{n-1}；而当前格子里麦子的数量就是新的变量值，我们记作X_{n}。这两个变量的递推关系就是这样的：



如果你稍微有点编程经验，应该能发现，迭代法的思想，很容易通过计算机语言中的**循环语言**来实现。你知道，计算机本身就适合做重复性的工作，我们可以通过循环语句，让计算机重复执行迭代中的递推步骤，然后推导出变量的最终值。

那接下来，我们就用循环语句来算算，填满格子到底需要多少粒麦子。我简单用Java语言写了个程序，你可以看看。

```
public class Lesson3_1 {
    /**
    * @Description: 算算舍罕王给了多少粒麦子
    * @param grid-放到第几格
    * @return long-麦粒的总数
    */

    public static long getNumberOfWheat(int grid) {

     long sum = 0;      // 麦粒总数
     long numberOfWheatInGrid = 0;  // 当前格子里麦粒的数量

     numberOfWheatInGrid = 1;  // 第一个格子里麦粒的数量
     sum += numberOfWheatInGrid;

     for (int i = 2; i <= grid; i ++) {
      numberOfWheatInGrid *= 2;   // 当前格子里麦粒的数量是前一格的2倍
      sum += numberOfWheatInGrid;   // 累计麦粒总数
     }

     return sum;

    }
}
```

下面是一段测试代码，它计算了到第63格时，总共需要多少麦粒。

```
  public static void main(String[] args) {
  System.out.println(String.format("舍罕王给了这么多粒：%d",   Lesson3_1.getNumberOfWheat(63)));
  }
```

计算的结果是9223372036854775807，多到数不清了。我大致估算了一下，一袋50斤的麦子估计有130万粒麦子，那么9223372036854775807相当于70949亿袋50斤的麦子！

这段代码有两个地方需要注意。首先，用于计算每格麦粒数的变量以及总麦粒数的变量都是Java中的long型，这是因为计算的结果实在是太大了，超出了Java int型的范围；第二，我们只计算到了第63格，这是因为计算到第64格之后，总数已经超过Java中long型的范围。

迭代法有什么具体应用?

看到这里，你可能大概已经理解迭代法的核心理念了。迭代法在无论是在数学，还是计算机领域都有很广泛的应用。大体上，迭代法可以运用在以下几个方面：

- **求数值的精确或者近似解**。典型的方法包括二分法（Bisection method）和牛顿迭代法（Newton’s method）。

- **在一定范围内查找目标值。**典型的方法包括二分查找。
- **机器学习算法中的迭代。**相关的算法或者模型有很多，比如K-均值算法（K-means clustering）、PageRank的马尔科夫链（Markov chain）、梯度下降法（Gradient descent）等等。迭代法之所以在机器学习中有广泛的应用，是因为**很多时候机器学习的过程，就是根据已知的数据和一定的假设，求一个局部最优解**。而迭代法可以帮助学习算法逐步搜索，直至发现这种解。

这里，我详细讲解一下求数值的解和查找匹配记录这两个应用。

1.求方程的精确或者近似解

迭代法在数学和编程的应用有很多，如果只能用来计算庞大的数字，那就太“暴殄天物”了。迭代还可以帮助我们进行无穷次地逼近，求得方程的精确或者近似解。

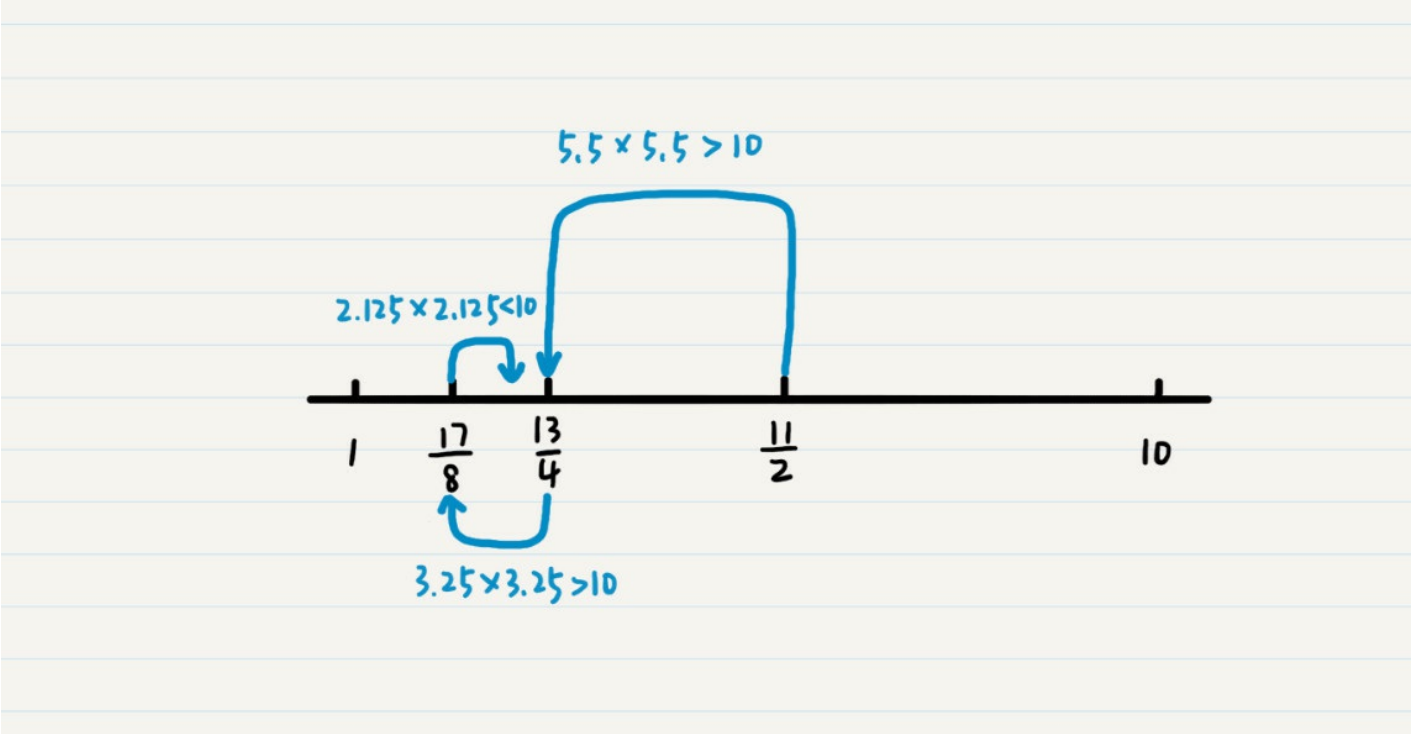
比如说，我们想计算某个给定正整数n（n>1）的平方根，如果不使用编程语言自带的函数，你会如何来实现呢?

假设有正整数n，这个平方根一定小于n本身，并且大于1。那么这个问题就转换成，在1到n之间，找一个数字等于n的平方根。

我这里采用迭代中常见的**二分法**。每次查看区间内的中间值，检验它是否符合标准。

举个例子，假如我们要找到10的平方根。我们需要先看1到10的中间数值，也就是11/2=5.5。5.5的平方是大于10的，所以我们要一个更小的数值，就看5.5和1之间的3.25。由于3.25的平方也是大于10的，继续查看3.25和1之间的数值，也就是2.125。这时，2.125的平方小于10了，所以看2.125和3.25之间的值，一直继续下去，直到发现某个数的平方正好是10。

我把具体的步骤画成了一张图，你可以看看。



我这里用Java代码演示一下效果，你可以结合上面的讲解，来理解迭代的过程。

```
public class Lesson3_2 {

 /**
    * @Description: 计算大于1的正整数之平方根
    * @param n-待求的数, deltaThreshold-误差的阈值, maxTry-二分查找的最大次数
    * @return double-平方根的解
    */
    public static double getSqureRoot(int n, double deltaThreshold, int maxTry) {

     if (n <= 1) {
      return -1.0;
     }

     double min = 1.0, max = (double)n;
     for (int i = 0; i < maxTry; i++) {
      double middle = (min + max) / 2;
      double square = middle * middle;
      double delta = Math.abs((square / n) - 1);
      if (delta <= deltaThreshold) {
       return middle;
      } else {
       if (square > n) {
        max = middle;
       } else {
        min = middle;
       }
      }
     }

     return -2.0;

    }
}
```

这是一段测试代码，我们用它来找正整数10的平方根。如果找不到精确解，我们就返回一个近似解。

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

  int number = 10;
  double squareRoot = Lesson3_2.getSqureRoot(number, 0.000001, 10000);
  if (squareRoot == -1.0) {
   System.out.println("请输入大于1的整数");
  } else if (squareRoot == -2.0) {
   System.out.println("未能找到解");
  } else {
   System.out.println(String.format("%d的平方根是%f", number, squareRoot));
  }

 }
```

这段代码的实现思想就是我前面讲的迭代过程，这里面有两个小细节我解释下。

第一，我使用了deltaThreshold来控制解的精度。虽然理论上来说，可以通过二分的无限次迭代求得精确解，但是考虑到实际应用中耗费的大量时间和计算资源，绝大部分情况下，我们并不需要完全精确的数据。

第二，我使用了maxTry来控制循环的次数。之所以没有使用while(true)循环，是为了避免死循环。虽然，在这里使用deltaThreshold，理论上是不会陷入死循环的，但是出于良好的编程习惯，我们还是尽量避免产生的可能性。

说完了二分迭代法，我这里再简单提一下牛顿迭代法。这是牛顿在17世纪提出的一种方法，用于求方程的近似解。这种方法以微分为基础，每次迭代的时候，它都会去找到比上一个值x_{0}更接近的方程的根，最终找到近似解。该方法及其延伸也被应用在机器学习的算法中，在之后机器学习中的应用中，我会具体介绍这个算法。

2.查找匹配记录

二分法中的迭代式逼近，不仅可以帮我们求得近似解，还可以帮助我们查找匹配的记录。我这里用一个查字典的案例来说明。

在自然语言处理中，我们经常要处理同义词或者近义词的扩展。这时，你手头上会有一个同义词/近义词的词典。对于一个待查找的单词，我们需要在字典中找出这个单词，以及它所对应的同义词和近义词，然后进行扩展。比如说，这个字典里有一个关于“西红柿”的词条，其同义词包括了“番茄”和“tomato”。

那么，在处理文章的时候，当我们看到了“西红柿”这个词，就去字典里查一把，拿出“番茄”“tomato”等等，并添加到文章中作为同义词/近义词的扩展。这样的话，用户在搜索“西红柿”这个词的时候，我们就能确保出现“番茄”或者“tomato”的文章会被返回

给用户。

乍一看到这个任务的时候，你也许想到了哈希表。没错，哈希表是个好方法。不过，如果不使用哈希表，你还有什么其他方法呢？这里，我来介绍一下，用二分查找法进行字典查询的思路。

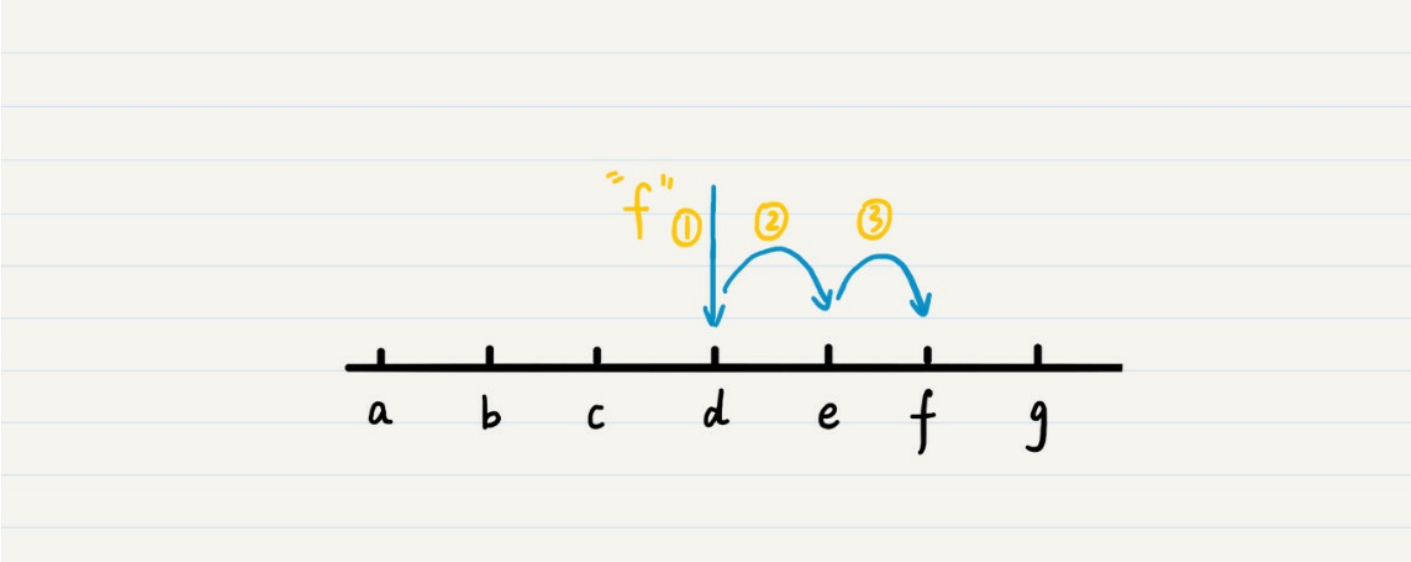
第一步，将整个字典先进行排序（假设从小到大）。二分法中很关键的前提条件是，所查找的区间是有序的。这样才能在每次折半的时候，确定被查找的对象属于左半边还是右半边。

第二步，使用二分法逐步定位到被查找的单词。每次迭代的时候，都找到被搜索区间的中间点，看看这个点上的单词，是否和待查单词一致。如果一致就返回；如果不一致，要看被查单词比中间点上的单词是小还是大。如果小，那说明被查的单词如果存在字典中，那一定在左半边；否则就在右半边。

第三步，根据第二步的判断，选择左半边或者后半边，继续迭代式地查找，直到范围缩小到单个的词。如果到最终仍然无法找到，则返回不存在。

当然，你也可以对单词进行从大到小的排序，如果是那样，在第二步的判断就需要相应地修改一下。

我把在a到g的7个字符中查找f的过程，画成了一张图，你可以看看。



这个方法的整体思路和二分法求解平方根是一致的，主要区别有两个方面：第一，每次判断是否终结迭代的条件不同。求平方根的时候，我们需要判断某个数的平方是否和输入的数据一致。而这里，我们需要判断字典中某个单词是否和待查的单词相同。第二，二分查找需要确保被搜索的空间是有序的。

我把具体的代码写出来了，你可以看一下。

```
import java.util.Arrays;

public class Lesson3_3 {

 /**
    * @Description: 查找某个单词是否在字典里出现
    * @param dictionary-排序后的字典, wordToFind-待查的单词
    * @return boolean-是否发现待查的单词
    */
    public static boolean search(String[] dictionary, String wordToFind) {

     if (dictionary == null) {
      return false;
     }

     if (dictionary.length == 0) {
      return false;
     }

     int left = 0, right = dictionary.length - 1;
     while (left <= right) {
      int middle = (left + right) / 2;
      if (dictionary[middle].equals(wordToFind)) {
       return true;
      } else {
       if (dictionary[middle].compareTo(wordToFind) > 0) {
        right = middle - 1;
       } else {
        left = middle + 1;
       }
      }
     }

     return false;

    }

}
```

我测试代码首先建立了一个非常简单的字典，然后使用二分查找法在这个字典中查找单词“i”。

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


  String[] dictionary = {"i", "am", "one", "of", "the", "authors", "in", "geekbang"};

  Arrays.sort(dictionary);

  String wordToFind = "i";

  boolean found = Lesson3_3.search(dictionary, wordToFind);
  if (found) {
   System.out.println(String.format("找到了单词%s", wordToFind));
  } else {
   System.out.println(String.format("未能找到单词%s", wordToFind));
  }

 }
```

说的这两个例子，都属于迭代法中的二分法，我在第一节的时候说过，二分法其实也体现了二进制的思想。

小结

到这里，我想你对迭代的核心思路有了比较深入的理解。

实际上，人类并不擅长重复性的劳动，而计算机却很适合做这种事。这也是为什么，以重复为特点的迭代法在编程中有着广泛的应用。不过，日常的实际项目可能并没有体现出明显的重复性，以至于让我们很容易就忽视了迭代法的使用。所以，你要多观察问题的现象，思考其本质，看看不断更新变量值或者缩小搜索的区间范围，是否可以获得最终的解（或近似解、局部最优解），如果是，那么你就可以尝试迭代法。

今日学习笔记

第3节　迭代法

1. 什么是迭代法？

迭代法，其实就是不断地用旧的变量值，递推计算新的变量值。迭代法的思想很容易通过计算机语言中的循环语言来实现。我们可以通过循环语句，让计算机重复执行迭代中的递推步骤，推导出变量的最终值。

2. 迭代法的基本步骤是什么？

确定用于迭代的变量。

建立迭代变量之间的递推关系。

控制迭代的过程。

3. 迭代法有什么具体应用？

求数值的精确或者近似解。

在一定范围内查找目标值。

机器学习算法中的迭代。

黄申 · 程序员的数学基础课

思考题

在你曾经做过的项目中，是否使用过迭代法？如果有，你觉得迭代法最大的特点是什么？如果还没用过，你想想看现在的项目中是否有可以使用的地方？

欢迎在留言区交作业，并写下你今天的学习笔记。你可以点击“请朋友读”，把今天的内容分享给你的好友，和他一起精进。

精选留言

云儿-199412
求一个数的平方根的那段代码中的第18行（double delta = Math.abs((square / n) - 1);）不太能看明白，为什么这么做？老师和专栏朋友们可以帮忙解决一下吗？谢谢。
2018-12-14 10:19

作者回复
我这里使用了误差占原值的百分比，来控制迭代的结束
2018-12-14 11:32

唐瑞甫
class Lesson3_3里面第22行改成 int middle = left + (right - left)/2 会更合适一点，不然有可能会溢出
2018-12-14 08:49

作者回复
对 很好的补充
2018-12-14 09:19

WL
没太看懂怎么用二分法查找同义词, 文章中讲的算法好像用二分法查询指定的单词, 不知道我这么理解对不对
2018-12-14 07:38

作者回复
对 其实是精确匹配，匹配后就可以拿到这个词对应的同义或近义词
2018-12-14 09:22

Jerry银银
老师，心里有点疑惑：感觉迭代法、数学归纳法有相关性，而且跟编程里面的循环和递归都有相关，您能否简要概括一下他们之间关系和联系呢？
2018-12-14 13:12

作者回复

这是个很好的问题，确实有些地方让人容易糊涂。我这里谈谈自己的理解。

数学里的迭代法，最初是用来求解方程的根，通过不断的更新变量值来逼近最终的解。其思想也被用来计算数列、二分查找等等。我把这种迭代法称为广义的。

而数学归纳法呢，是从理论上证明某个命题成立，从而避免了迭代中的重复计算。下一篇会具体介绍。

而递归就是指“递推”和“回归”，它的递推和数学归纳法非常类似，因此数学归纳法中的递推可以直接翻译为递归的编程。而循环也有递推，不过通常和递归是反向的。

此外，人们常常把编程中的基于循环的实现叫做迭代的实现，用于和递归的实现加以区分。我个人觉得这种迭代的叫法是狭义的。广义的迭代既可以使用循环，也可以使用递归来实现，就像我第3讲的求根和二分查找等，也可以用递归来实现。

2018-12-14 15:08

晓嘿

老师

“唐瑞甫

[?]

2

class Lesson3_3里面第22行改成 int middle = left + (right - left)/2 会更合适一点，不然有可能会溢出

2018-12-14

[?] 作者回复

对 很好的补充”

这个我看着跟你写的那个是一样的啊，换算完也是(left+right)/2啊，这个22行的代码会溢出吗，在什么情况下

2018-12-14 11:24

作者回复

确实从数学的角度看是一样的，但是计算机系统本身有局限性。如果left和right都是接近系统设定的最大值，那么两者相加会溢出。如果只加两者差值的一半，那么不会超过两者中较大的值，自然也不会溢出

2018-12-14 14:07

柚子

程序论递归和迭代区别，突然有个想法，好像将结束条件写在方法里就是递归，将结束条件写在方法外就是迭代。哈哈

2018-12-14 20:44

作者回复

在编程里，递归的主要特征是方法或函数自己调用自己，因此一般结束条件放在方法内。而基于循环的迭代，如果递推是方法实现的，那确实结束条件是在方法外

2018-12-14 23:06

Wing·三金

目前正在做机器学习最优化方面的研究，所以对迭代法应用很多，几乎可以说是科研人员的必备手段了。

迭代法最困难的地方除了设置「迭代的规则」，另一个难点就是设置「迭代的终止条件」。前者专业性比较强就不多说，后者很大程度上依赖于coder的经验。因为机器学习中往往只要求足够精确的近似解，而如果一昧追求精度可能时间复杂度太大；如果以最大迭代次数为终止条件又可能得不到满意的解。因此实践中往往二者一起用，而且精度和迭代次数都需要根据一定的理论支撑去设定（不过更多的时候是从业界认可的经验出发）。

2018-12-15 17:55

作者回复

很好的心得体会

2018-12-15 23:44

瘦马

迭代的基本思想就是不断用旧的变量推算出新的变量，直到获得有效结果。

迭代使用的步骤：

1、确定变量

2、确定变量的推导方式

3、控制迭代

2018-12-14 13:16

silence

迭代就是将问题相同的部分抽离出来，把不容易解决的大问题切割成一个个小问题

2018-12-14 19:54

作者回复

递归式的迭代可以将大问题逐步简化为小问题

2018-12-14 23:08

彩色的沙漠

快速排序，用的也是二分迭代思想，把一个数组分成两个独立部分。分别进行排序，直到两边都是有序

2018-12-19 20:21

作者回复

是的，采用了分而治之的思想

2018-12-19 22:47

代码世界没有爱情

python实现：

```
def f(x):
y = x
if x > 1 and isinstance(x, int):
flag, num = 1, 0
global middle
while num <= 100:

middle = (x + flag) / 2
if middle * middle > y:
x = middle
elif middle * middle < y:
flag = middle
else:
print('exactly value:', middle)

break
num += 1
else:
print('deferenct value:', middle)
elif x ==1:print('exactly value:', 1)
else:
print('TypeError')

f(81)
```

2018-12-17 15:38

我不是王子

老师，求平方根的第18行我也没看懂，可以详细讲解一下吗，为什么是(square / n) - 1再求绝对值呢

2018-12-15 11:49

作者回复

这是算相对误差，比如n是100，那么误差为1的时候，误差相对于n的百分比为1%。

2018-12-15 14:58

指间砂的宿命

二分法很少手写，程序中更多使用循环语句，不过对于有序数据查找二分法倒是相对高效，工作中倒是很少用，特别是有数据库的情况下指定key很多时候都是直接让数据库返回了

2018-12-14 09:33

作者回复

有些数据库的索引，具体实现的时候可能会用到二分查找

2018-12-14 09:42

M

老师可以用伪代码写么，没学过JAVA..

2018-12-16 10:12

作者回复

你学得什么语言?

2018-12-16 13:04

来碗绿豆汤

找词那个，我通过西红柿可以找到番茄和tomato,但是怎么通过tomato找到西红柿呢

2018-12-14 22:58

作者回复

需要加一个以tomato为key的词条

2018-12-15 00:12

（+曦+）

7皇后问题

2018-12-14 20:03

liujingang

为什么是循环而不是递归呢?

2018-12-14 09:19

作者回复

二分即可以用循环 也可以用递归

2018-12-14 09:40

asc

不理解同义词跟二分法查找合体的意义，老师是想表达所有的同义词都在字典里面找？不管用户给的番茄还是西红柿都能指向到西红柿这个词吗

2018-12-14 08:33

作者回复

对 其实二分就是查到“西红柿”这条记录，然后再看其对应的同义或近义词。二分查找相当于使用哈希表里的key找到对应的entry，然后就能拿到entry里的value，也就是同义或近义词

2018-12-14 09:25

一颗菜

印度国王的例子就是巴菲特的复利

2018-12-14 08:22

牛玉富

老师第二个例子只能查找一个单词，建议修改成多次支持的形式

2019-01-09 10:07

作者回复

这里多次支持是指多个单词的查询?

2019-01-09 23:33