

03 | 迭代法：不用编程语言的自带函数，你会如何计算平方根？

2018-12-14 黄申

程序员的数学基础课

[进入课程 >](#)



讲述：黄申

时长 11:38 大小 10.67M



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：

The diagram shows the iterative formula $f(X_n) = f(X_{n-1}) * 2$ and the initial condition $f(1) = 1$. Annotations include: "(旧变量值)" (Old variable value) pointing to X_{n-1} , "前一格的麦粒数量" (Wheat grain quantity of the previous square) pointing to $f(X_{n-1})$, "(新变量值)" (New variable value) pointing to X_n , and "当前格子中的麦粒数量" (Wheat grain quantity in the current square) pointing to $f(X_n)$.

$$f(X_n) = f(X_{n-1}) * 2$$
$$f(1) = 1$$

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。

 复制代码

```
1 public class Lesson3_1 {
2     /**
3      * @Description: 算算舍罕王给了多少粒麦子
4      * @param grid- 放到第几格
5      * @return long- 麦粒的总数
6      */
7
8     public static long getNumberOfWheat(int grid) {
9
10        long sum = 0;        // 麦粒总数
11        long numberOfWheatInGrid = 0; // 当前格子里麦粒的数量
12
13        numberOfWheatInGrid = 1; // 第一个格子里麦粒的数量
14        sum += numberOfWheatInGrid;
15
16        for (int i = 2; i <= grid; i++) {
17            numberOfWheatInGrid *= 2; // 当前格子里麦粒的数量是前一格的 2 倍
18            sum += numberOfWheatInGrid; // 累计麦粒总数
19        }
20
21        return sum;
22
23    }
24 }
```

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。

 复制代码

```
1 public static void main(String[] args) {
2     System.out.println(String.format(" 舍罕王给了这么多粒: %d", Lesson3_1.getNumberOfWhea
3 }
```

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。

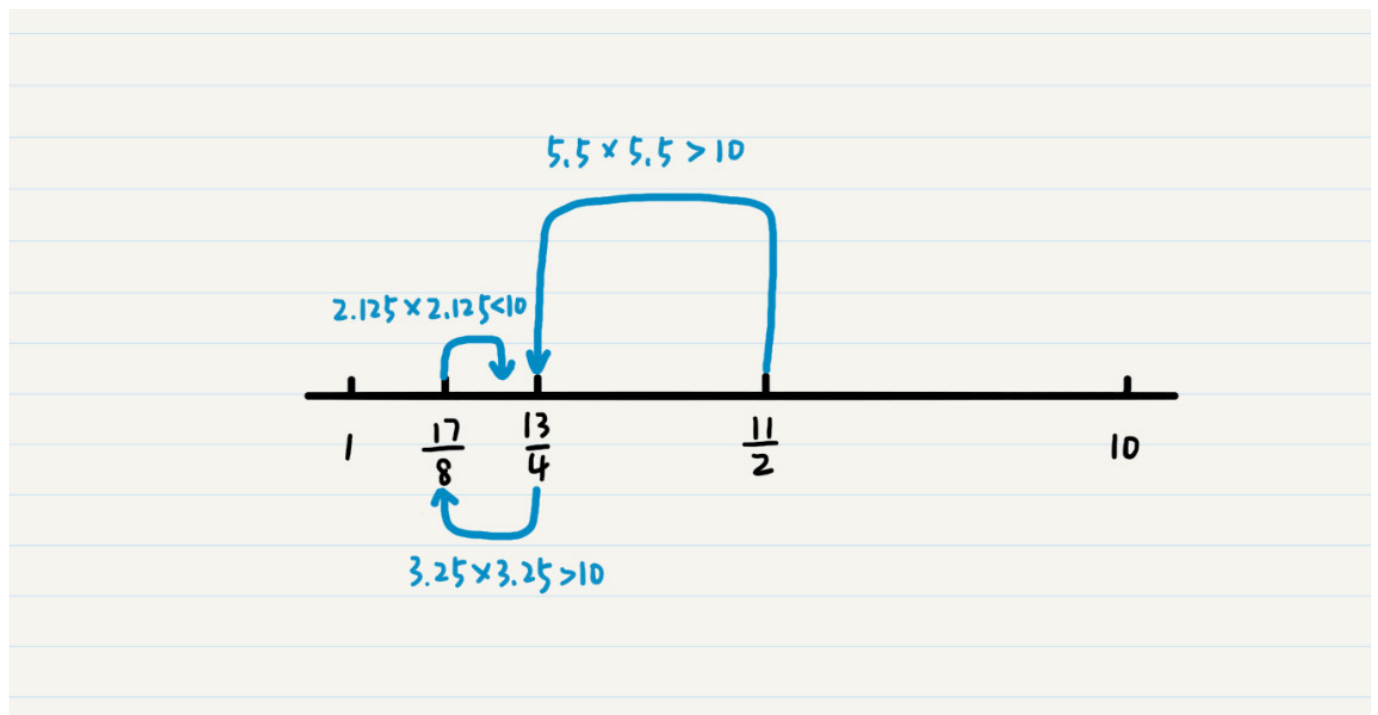
比如说，我们想计算某个给定正整数 n ($n > 1$) 的平方根，如果不使用编程语言自带的函数，你会如何来实现呢？

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。

 复制代码


```
1 public class Lesson3_2 {
2
3     /**
4      * @Description: 计算大于 1 的正整数之平方根
5      * @param n- 待求的数, deltaThreshold- 误差的阈值, maxTry- 二分查找的最大次数
6      * @return double- 平方根的解
7      */
8     public static double getSquireRoot(int n, double deltaThreshold, int maxTry) {
9
10         if (n <= 1) {
11             return -1.0;
12         }
13
14         double min = 1.0, max = (double)n;
15         for (int i = 0; i < maxTry; i++) {
16             double middle = (min + max) / 2;
```

```

17     double square = middle * middle;
18     double delta = Math.abs((square / n) - 1);
19     if (delta <= deltaThreshold) {
20         return middle;
21     } else {
22         if (square > n) {
23             max = middle;
24         } else {
25             min = middle;
26         }
27     }
28 }
29
30 return -2.0;
31
32 }
33 }

```

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。

 复制代码

```

1 public static void main(String[] args) {
2
3     int number = 10;
4     double squareRoot = Lesson3_2.getSquireRoot(number, 0.000001, 10000);
5     if (squareRoot == -1.0) {
6         System.out.println(" 请输入大于 1 的整数 ");
7     } else if (squareRoot == -2.0) {
8         System.out.println(" 未能找到解 ");
9     } else {
10        System.out.println(String.format("%d 的平方根是 %f", number, squareRoot));
11    }
12
13 }

```

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”。

词条	同义词1	同义词2	同义词3
西红柿	番茄	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。

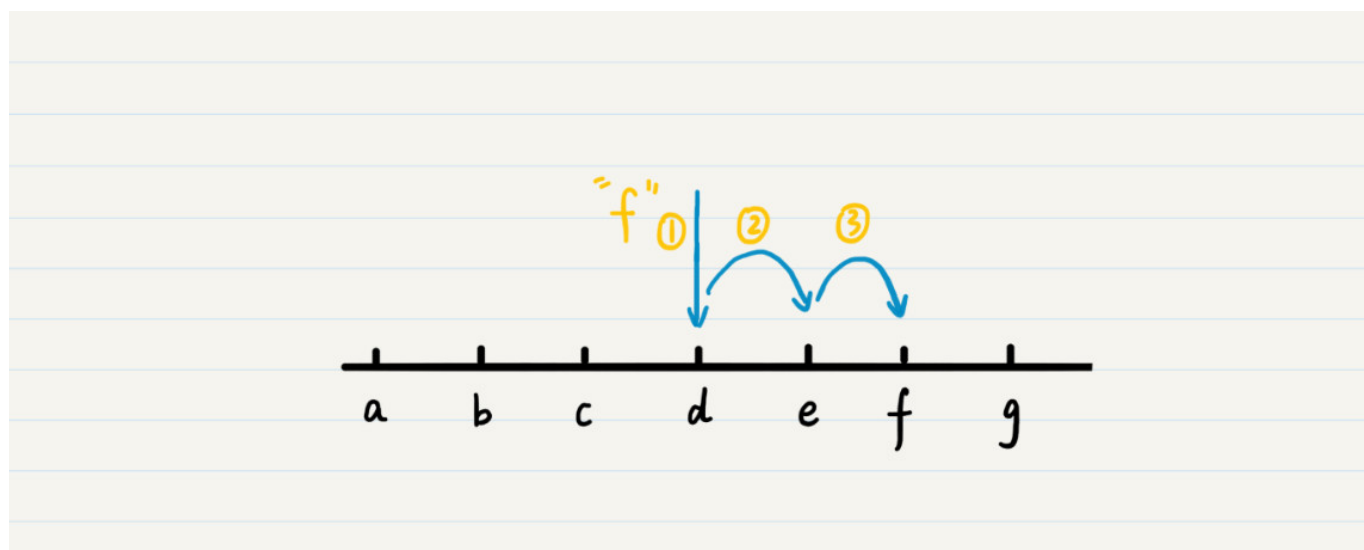
第一步，将整个字典先进行排序（假设从小到大）。二分法中很关键的前提条件是，所查找的区间是有序的。这样才能在每次折半的时候，确定被查找的对象属于左半边还是右半边。

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，选择左半边或者右半边，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。

 复制代码

```
1 import java.util.Arrays;
```




```

2
3 public class Lesson3_3 {
4
5     /**
6      * @Description: 查找某个单词是否在字典里出现
7      * @param dictionary- 排序后的字典, wordToFind- 待查的单词
8      * @return boolean- 是否发现待查的单词
9      */
10    public static boolean search(String[] dictionary, String wordToFind) {
11
12        if (dictionary == null) {
13            return false;
14        }
15
16        if (dictionary.length == 0) {
17            return false;
18        }
19
20        int left = 0, right = dictionary.length - 1;
21        while (left <= right) {
22            int middle = (left + right) / 2;
23            if (dictionary[middle].equals(wordToFind)) {
24                return true;
25            } else {
26                if (dictionary[middle].compareTo(wordToFind) > 0) {
27                    right = middle - 1;
28                } else {
29                    left = middle + 1;
30                }
31            }
32        }
33
34        return false;
35
36    }
37
38 }
39
40

```

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” 。

 复制代码

```

1 public static void main(String[] args) {
2
3

```

```
4 String[] dictionary = {"i", "am", "one", "of", "the", "authors", "in", "geekbang"};
5
6 Arrays.sort(dictionary);
7
8 String wordToFind = "i";
9
10 boolean found = Lesson3_3.search(dictionary, wordToFind);
11 if (found) {
12     System.out.println(String.format(" 找到了单词 %s", wordToFind));
13 } else {
14     System.out.println(String.format(" 未能找到单词 %s", wordToFind));
15 }
16
17 }
```

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。



程序员的数学基础课

在实战中重新理解数学

黄申

LinkedIn 资深数据科学家



新版升级：点击「👤 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 02 | 余数：原来取余操作本身就是个哈希函数

下一篇 04 | 数学归纳法：如何用数学归纳提升代码的运行效率？

精选留言 (71)

写留言



云儿-1994... 置顶

2018-12-14

23

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。

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





唐瑞甫

2018-12-14

👍 52

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

作者回复: 对 很好的补充



晓嘿

2018-12-14

👍 25

老师

“唐瑞甫

□

2

class Lesson3_3里面第22行改成 $\text{int middle} = \text{left} + (\text{right} - \text{left})/2$ 会更合适一点，不...

展开 ▾

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



Jerry银银

2018-12-14

👍 21

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？

展开 ▾

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。



WL

2018-12-14

14

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

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



Wing•三金

2018-12-15

11

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...
展开

作者回复: 很好的心得体会



柚子

2018-12-14

9

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

展开

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



瘦马

2018-12-14

6

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

展开 ▾



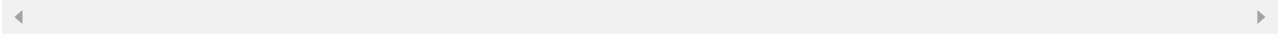
彩色的沙漠

2018-12-19

👍 4

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

作者回复: 是的，采用了分而治之的思想



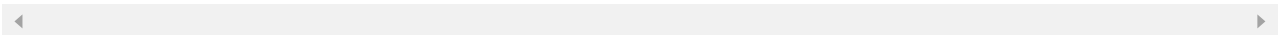
silence

2018-12-14

👍 4

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

作者回复: 递归式的迭代可以将大问题逐步简化为小问题



代码世界没...

2018-12-17

👍 3

python实现：

```
def f(x):
```

```
    y = x
```

```
    if x > 1 and isinstance(x, int):
```

```
        flag, num = 1, 0...
```

展开 ▾



蒋宏伟

2018-12-15

👍 3

迭代法

why

利用计算机适合重复计算的特点

how

$f(0)=...$

展开 ▾



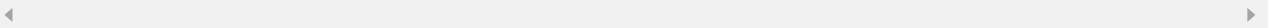
Shawn

2019-03-23

👍 2

既然提到了求平方根就不得不说一下神奇的魔术字:0x5f3759df

作者回复: 确实是个申请的数字, 还研究了好久背后的数学知识



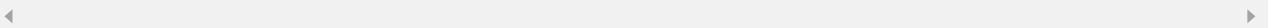
我不是王子

2018-12-15

👍 2

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 $(\text{square} / n) - 1$ 再求绝对值呢

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%。



(+曦+)

2018-12-14

👍 2

7皇后问题

展开 ▾



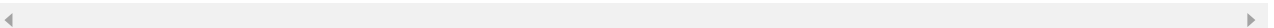
指间砂的宿...

2018-12-14

👍 2

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

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





Geek_zy

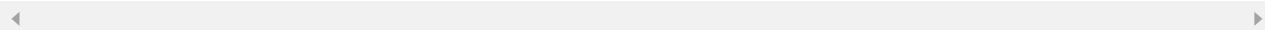
2019-04-25

👍 1

老师，之前看过递归，分治思想这些问题。所以就有复习了一下。得到一下总结：迭代是一种解决问题的思想，然后递归和循环都是实现这种思想的编程手段。包括分治思想也是他也是一种思想，正好迭代这种编程手段可以来实现这种思想。不知道我的理解有什么问题没有，望老师指教。

展开 ▾

作者回复: 理解是对的。数学里的迭代法更为通用，而编程里的“迭代法”一般特指基于循环的实现。



caohuan

2019-01-21

👍 1

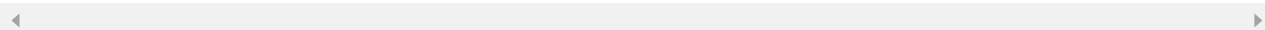
黄老师的数学课的应用 果然好看，必须常来。

本篇所得1.迭代法的思路已经在 算法得到很多的运用，机器学习中有体现，比如老师提到的：kmean、马尔可夫链、梯度下降等；

2.迭代法的特点是：下一个数据 与 上一个数据有固定的关系，可以应用 方程式表达。...

展开 ▾

作者回复: 后面会介绍迭代法和递归法的关系，在我看来数学中的迭代法相对于编程中的迭代法更广义。



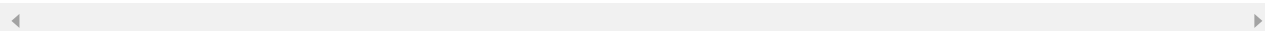
耿森

2018-12-29

👍 1

在贷款还款计算中，如果贷款方式是等额本金，那么每期的还款金额是根据上一期来计算的，要用到迭代法 😊

作者回复: 厉害了，非常好的生活实例



kylin

2018-12-17

👍 1

请问class3_3中的22行 $\text{int mid} = (\text{left} + \text{right}) / 2$ 为啥会可能有溢出，如果改成 int mid

= $\text{left} + (\text{right} - \text{left}) / 2$ 就不会溢出了呢？

展开 ∨

作者回复: 理论上两者一样。主要是当left和right都趋近于计算机系统设定的最大值时，就可能溢出，你可以画个图试试看

