**抽象分析模式**

**抽象分析模式，实际上是一种寻找问题解决方向或突破口的思维方法，算法面试中，最怕的是感觉无处着手，本章所要探寻的，是告诉大家，拿到问题后，我们怎么想才能更好更快的分析出问题的着眼点。**

**样例分析**

**样例分析是指将问题分解成若干种样例，然后对每种样例情况进行分析，当所有样例都分析完毕后，问题自然就得以解决。样例的选择不需要很多，但要尽可能的覆盖问题的领域。举个例子，证明 n^3 mod 3 = 0， 1， 8. 面对这个问题，我们可以把n 分解成三种样例，分别是n = 3m, n = 3m + 1, n = 3m + 2. 每一种样例都很容易证明所给结论，而n取值也就只能是三种样例之一，所以，三种样例都能满足证明，那么n就能满足证明。**

**小数量数据分析**

**这种方法是算法面试中，最常用，也是最有效的办法。如果我们直接从逻辑上对问题进行推导感觉很困难的时候，我们可以设计一下数量小的实例数据带入到问题中，看看能否得到有效的思路。举个例子，在走廊上有五百扇门，编号分别是1到500.第一个人走过走廊，然后打开每一扇门，第二个人走过走廊，把编号为2，4，6… 的门都给关上，就这样，第i个人经过时，把编号为i, 2\*i, 3\*i, …. 的门的状态取反，也就是如果门是关的，那就打开，如果是打开的，那就关上。试问第500个人后，有多少扇门是开着的。这个问题，如果直接进行推导，要得出答案是很困难的，但如果我们用一些小数量的实例来分析，很容易得到有效的线索。例如，我们看看，如果门的数量分别是1，2，3，4 ， 10 和20时，我们能容易的看到，经过题目所说的操作后，最后门开着的数量分别是1， 4， 9， 16…, 不管门的总共数量是多少，最终开着的门的数量总保持上面的规律。也就是开着门的数量是最接近于总数量的完全平方数。那么如果总共是500扇门，那么500开平方是22，22的平方是484，也就是最接近500的完全平方数是484，因此我们得出总是为500扇门的话，最后开着门的数量是484.**

**简单方案的逐步改进**

**很多面试算法题，很多时候，是容易找到一种简单但是时间效率不好的算法。例如前几章反复强调的暴力枚举法就是一例。面对一个问题，我们可以先找出简单但时间效率差的算法，然后再在该基础上不断改进。**

**举个例子，给你一个含有n个元素的数组，让你写一个函数，将A中的元素从新排列后得到新的数组B,使得B有以下特点：**

**B[0] <= B[1] >= B[2] <= B[3] >= B[4] <= B[5]…..**

**一种简单但是效率不高的办法是先将A降序排序，然后对排序后的元素两两交换位置，假定A的元素如下：**

**1 3 4 5 8 6 2 7 9 10**

**降序排列后：**

**10 9 8 7 6 5 4 3 2 1**

**两两交换位置：**

**9 10 7 8 5 6 3 5 1 2**

**排序的时间复杂度是o(n lg n). 所以上面算法的时间复杂度是o(n lg n).**

**我们能不能在这个基础上进行改进呢？如果我们能找到数组的中位数，（中位数是指数组排序后，正好处于中间的那个元素），然后把数组元素分成两部分，一部分小于中位数，一部分大于中位数，然后将两部分的元素交叉组合就可以了，以上面例子为例， A 的中位数是5，于是数组分成两部分：**

**1 3 4 2 5 和 8 6 7 9 10，然后1 和 8 配对，3 和 6配对，这就有：1 8 3 6 4 7 2 9 5 10**

**由于查找中位数的时间复杂度是O(n)(查找中位数的算法我们会在后面详细介绍), 因此总体复杂度是O(n).**

**还有没有更好的办法呢，如果我们将数组遍历一次，如果i是偶数，且A[i] > A[i+1], 那么就交换A[i], A[i+1], 如果i是奇数，且A[i] < A[i+1], 那就交换A[i], A[i+1], 这么做也能达到最终效果，虽然时间复杂度跟上一种方案一样，但它简单，代码容易写：**

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

**if ((i % 2 ==0 && A[i] < A[i+1]) || (i % 2 == 1 && A[i] > A[i+1])) {**

**swap(A[i], A[i+1]);**

**}**

**}**

**问题还原**

**面试算法题中，不少情况下会问到这样一个问题，给定两个个字符串，判断第一个字符串是否是第二个个字符串的倒转，例如 “car” 的倒转可以是”arc”, “rca”, 一种简单的算法是，将原字符串在每个字符位置开始倒转，然后把倒转后的字符串跟给定字符串匹配，这么做的话，时间复杂度是o(n^2). 这个问题还原其本质其实是字符串匹配，也就是给定字符串s,判断s是否是字符串T的子串，字符串匹配有复杂度为o(n)的算法，如果我们把给定字符串首尾相连，然后判断第一个字符串是否是第二个字符串首尾相连后的子串，是的话，那么第二个字符串就是第一个字符串的倒转。例如”arc” 首尾相连后 是 “arcarc”, 而”car”是它的子串，所以”arc”是”car”的倒转，这样我们就相当于把问题还原为字符串匹配的问题，从而得到一个时间复杂度更好的解法。**

**图论模拟**

**画图是进行推导或思考的很好的辅助方式。同理，如果给定的问题能够转换为图论，那么我们可以套用图论的算法解决当前问题。例如，给定一组货币的相互转换汇率，然后让你决定是否存在汇率套利，也就是，你选定某一种货币，然后经过一系列的货币交易，最后换回到原来的货币时，得到的币值比原来更高。**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **USD** | **EUR** | **GBP** | **JPY** | **CHF** | **CAD** | **AUD** |
| **USD** | **1** | **0.8148** | **0.6404** | **78.125** | **0.9784** | **0.9924** | **0.9645** |
| **EUR** | **1.2275** | **1** | **0.7860** | **96.55** | **1.2010** | **1.2182** | **1.1616** |
| **GBP** | **1.5617** | **1.2724** | **1** | **122.83** | **1.5280** | **1.5498** | **1.4778** |
| **JPY** | **0.0128** | **1.0.0104** | **0.0081** | **1** | **1.2442** | **0.0126** | **0.0120** |
| **CHF** | **1.0219** | **0.8327** | **0.6546** | **80.39** | **1** | **1.0142** | **0.9672** |
| **CAD** | **1.0076** | **0.8206** | **0.6453** | **79.26** | **0.9859** | **1** | **0.9535** |
| **AUD** | **1.0567** | **0.8609** | **0.6767** | **83.12** | **1.0039** | **1.0487** | **1** |

**根据上面汇率表，我们有一条套利路径，那就是 1 USD -> 1\* 0.8213 = 0.8213EUR - > 0.8213 \* 1.2010 = 0.9755723CHF -> 0.9755723 \* 80.39 = 78.426257197JPY -> 78.426257197 \* 0.0128 = 1. 00385609212USD**

**这个问题我们可以转换成图论，每一种货币对应于图的一个节点，货币间的相互汇率取对数（lg）作为点和点间边的距离，于是问题转换为是否找到一个环，使得环的距离之和为正数。这样的环可以通过Bellman-Ford算法找到，图论的具体内容我们后面会讲到。**

**当大家在面试时，如果一开始想不到突破口或着手点，不如将我们本节说的方法带入到问题中看看，一般而言，面试算法题的难度有限，因此题目的思路范围不会太宽，因此将我们说过的方法套入题目，得到有效线索的概率是非常大的。**