02丨数据结构原理：Hash表的时间复杂度为什么是O(1)?

2019-11-20 李智慧

后端技术基础详解 进入课程 >

讲述：李智慧
时长 12:54　大小 11.83M

大概十年前，我在阿里巴巴工作的时候，曾经和另一个面试官一起进行一场技术面试，面试过程中我问了一个问题：**Hash 表的时间复杂度为什么是 O(1)**？候选人没有回答上来。面试结束后我和另一个面试官有了分歧，我觉得这个问题没有回答上来是不可接受的。而他则觉得，这个问题有一点难度，回答不上来不说明什么。

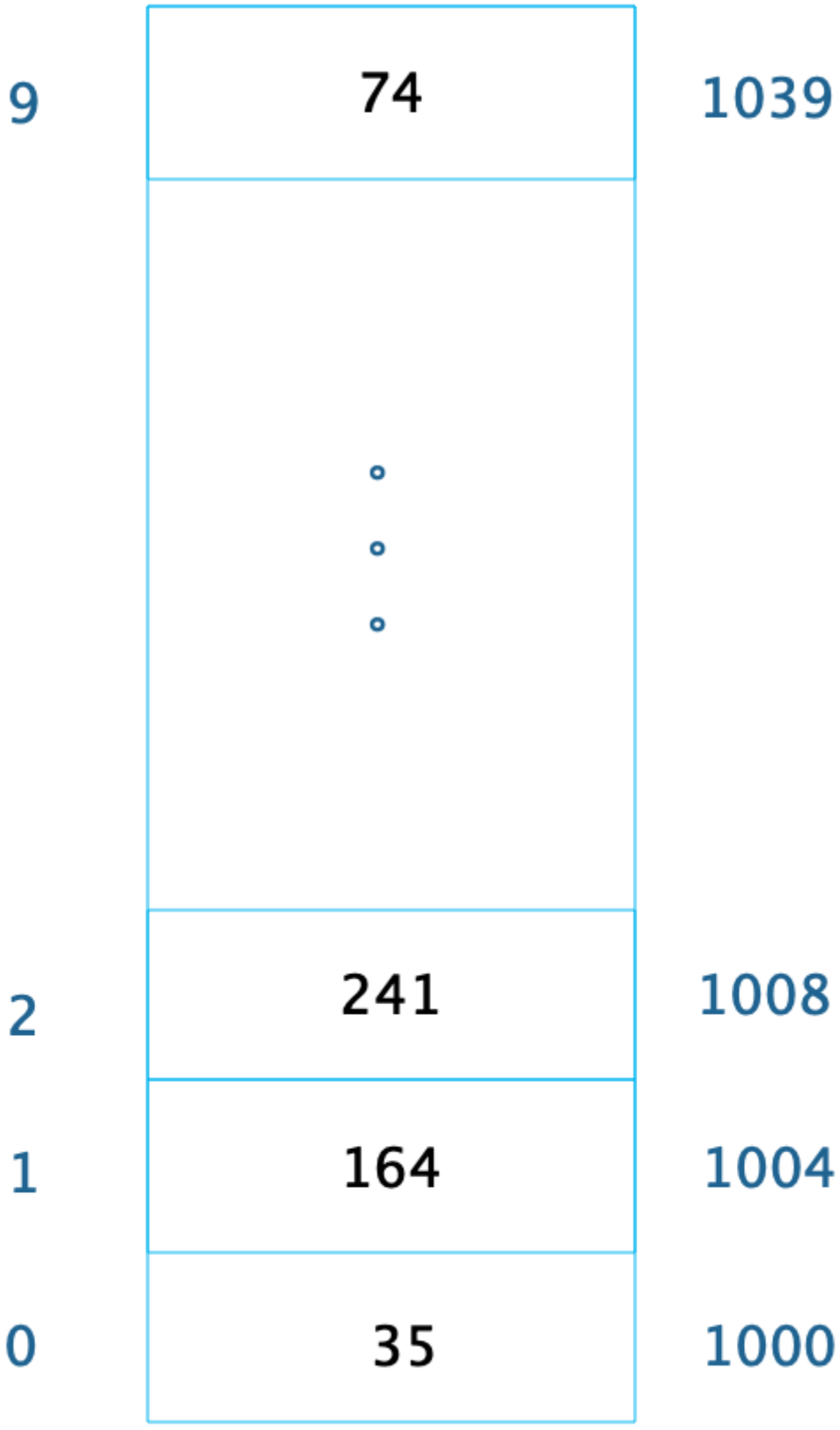
因为有了这次争执，后来这个问题成了我面试时的必考题。此后十年间，我用这个问题面试了大约上千人，这些面试经历让我更加坚定了一个想法：这个问题就是候选人技术水平的一个分水岭，是证明一个技术人员是否具有必备专业技能和技术悟性的一个门槛。这个槛过不去是不可接受的。

为什么呢？我很难相信，如果基本的数据结构没有掌握好，如何能开发好一个稍微复杂一点的程序？

要了解 Hash 表，需要先从数组说起。

数组

数组是最常用的数据结构，创建数组必须要内存中一块**连续**的空间，并且数组中必须存放**相同**的数据类型。比如我们创建一个长度为 10，数据类型为整型的数组，在内存中的地址是从 1000 开始，那么它在内存中的存储格式如下。



由于每个整型数据占据 4 个字节的内存空间，因此整个数组的内存空间地址是 1000～1039，根据这个，我们就可以轻易算出数组中每个数据的内存下标地址。利用这个特性，我们只要知道了数组下标，也就是数据在数组中的位置，比如下标 2，就可以计算得到这个数据在内存中的位置 1008，从而对这个位置的数据 241 进行快速读写访问，时间复杂度为 O(1)。

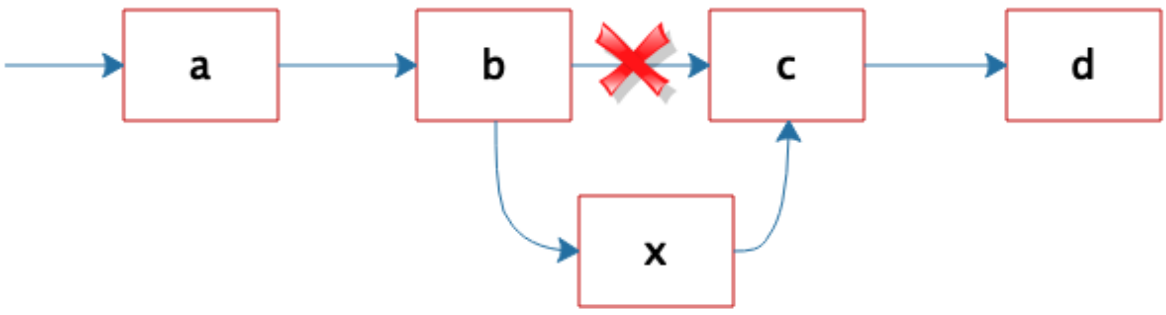
随机快速读写是数组的一个重要特性，但是要随机访问数据，必须知道数据在数组中的下标。如果我们只是知道数据的值，想要在数组中找到这个值，那么就只能遍历整个数组，时间复杂度为 O(N)。

链表

不同于数组必须要连续的内存空间，链表可以使用零散的内存空间存储数据。不过，因为链表在内存中的数据不是连续的，所以链表中的每个数据元素都必须包含一个指向下一个数据元素的内存地址指针。如下图，链表的每个元素包含两部分，一部分是数据，一部分是指向下一个元素的地址指针。最后一个元素指向 null，表示链表到此为止。

因为链表是不连续存储的，要想在链表中查找一个数据，只能遍历链表，所以链表的查找复杂度总是 O(N)。

但是正因为链表是不连续存储的，所以在链表中插入或者删除一个数据是非常容易的，只要找到要插入（删除）的位置，修改链表指针就可以了。如图，想在 b 和 c 之间插入一个元素 x，只需要将 b 指向 c 的指针修改为指向 x，然后将 x 的指针指向 c 就可以了。



相比在链表中轻易插入、删除一个元素这种简单的操作，如果我们要想在数组中插入、删除一个数据，就会改变数组连续内存空间的大小，需要重新分配内存空间，这样要复杂得多。

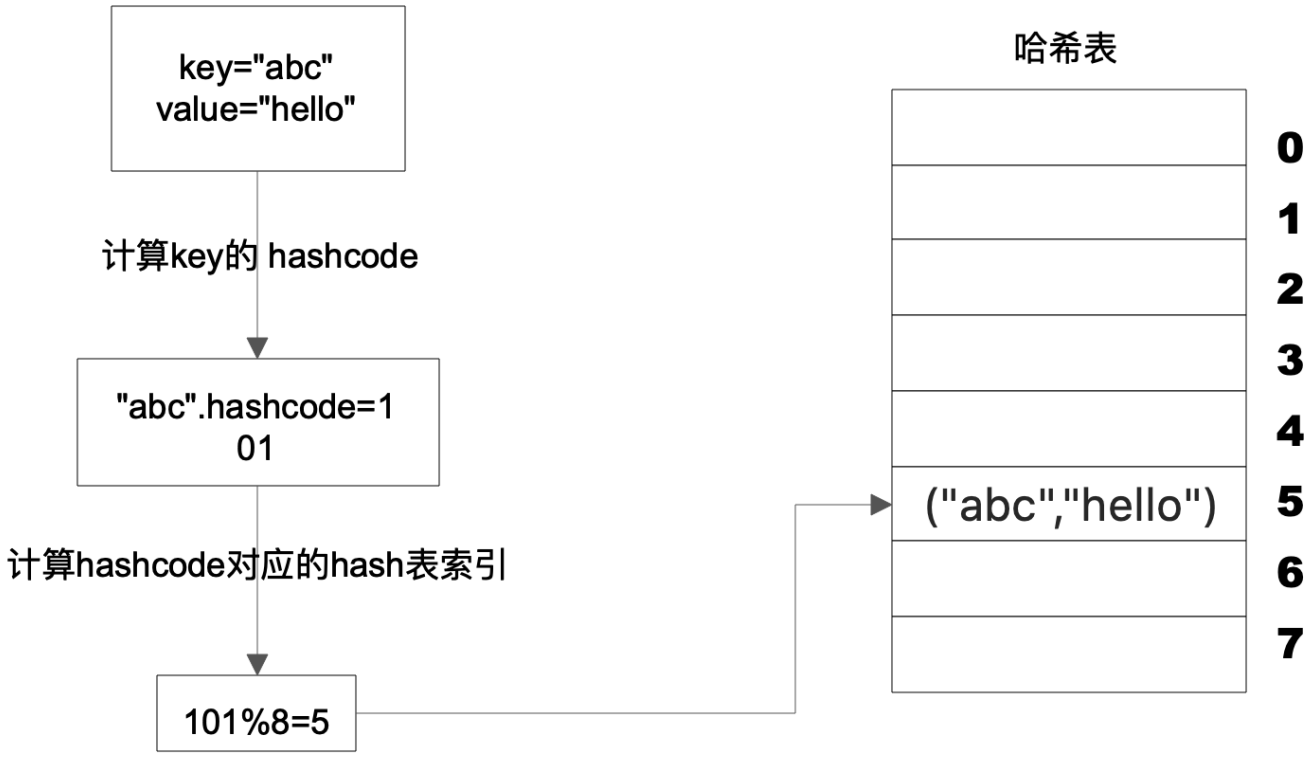
Hash 表

前面说过，对数组中的数据进行快速访问必须要通过数组的下标，时间复杂度为 O(1)。如果只知道数据或者数据中的部分内容，想在数组中找到这个数据，还是需要遍历数组，时间复杂度为 O(N)。

事实上，知道部分数据查找完整数据的需求在软件开发中会经常用到，比如知道了商品 ID，想要查找完整的商品信息；知道了词条名称，想要查找百科词条中的详细信息等。

这类场景就需要用到 Hash 表这种数据结构。Hash 表中数据以 Key、Value 的方式存储，上面例子中，商品 ID 和词条名称就是 Key，商品信息和词条详细信息就是 Value。存储的时候将 Key、Value 写入 Hash 表，读取的时候，只需要提供 Key，就可以快速查找到 Value。

Hash 表的物理存储其实是一个数组，如果我们能够根据 Key 计算出数组下标，那么就可以快速在数组中查找到需要的 Key 和 Value。许多编程语言支持获得任意对象的 HashCode，比如 Java 语言中 HashCode 方法包含在根对象 Object 中，其返回值是一个 Int。我们可以利用这个 Int 类型的 HashCode 计算数组下标。最简单的方法就是余数法，使用 Hash 表的数组长度对 HashCode 求余， 余数即为 Hash 表数组的下标，使用这个下标就可以直接访问得到 Hash 表中存储的 Key、Value。



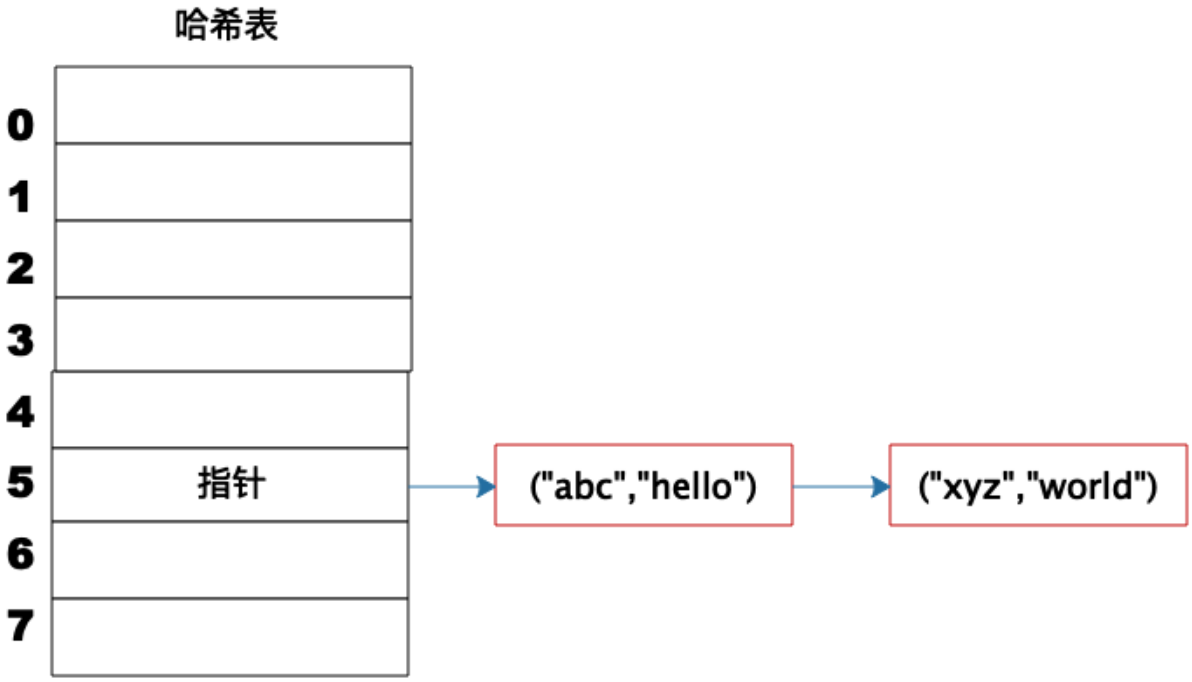
上图这个例子中，Key 是字符串 abc，Value 是字符串 hello。我们先计算 Key 的哈希值，得到 101 这样一个整型值。然后用 101 对 8 取模，这个 8 是哈希表数组的长度。101 对 8 取模余 5，这个 5 就是数组的下标，这样就可以把（“abc”，“hello”）这样一个 Key、Value 值存储在下标为 5 的数组记录中。

当我们要读取数据的时候，只要给定 Key abc，还是用这样一个算法过程，先求取它的 HashCode 101，然后再对 8 取模，因为数组的长度不变，对 8 取模以后依然是余 5，那么我们到数组下标中去找 5 的这个位置，就可以找到前面存储进去的 abc 对应的 Value 值。

但是如果不同的 Key 计算出来的数组下标相同怎么办？HashCode101 对 8 取模余数是 5，HashCode109 对 8 取模余数还是 5，也就是说，不同的 Key 有可能计算得到相同的数组下标，这就是所谓的 Hash 冲突，解决 Hash 冲突常用的方法是链表法。

事实上，（“abc”，“hello”）这样的 Key、Value 数据并不会直接存储在 Hash 表的数组中，因为数组要求存储固定数据类型，主要目的是每个数组元素中要存放固定长度的数据。所以，数组中存储的是 Key、Value 数据元素的地址指针。一旦发生 Hash 冲突，只需要将相同下标，不同 Key 的数据元素添加到这个链表就可以了。查找的时候再遍历这个链表，匹配正确的 Key。

如下图：

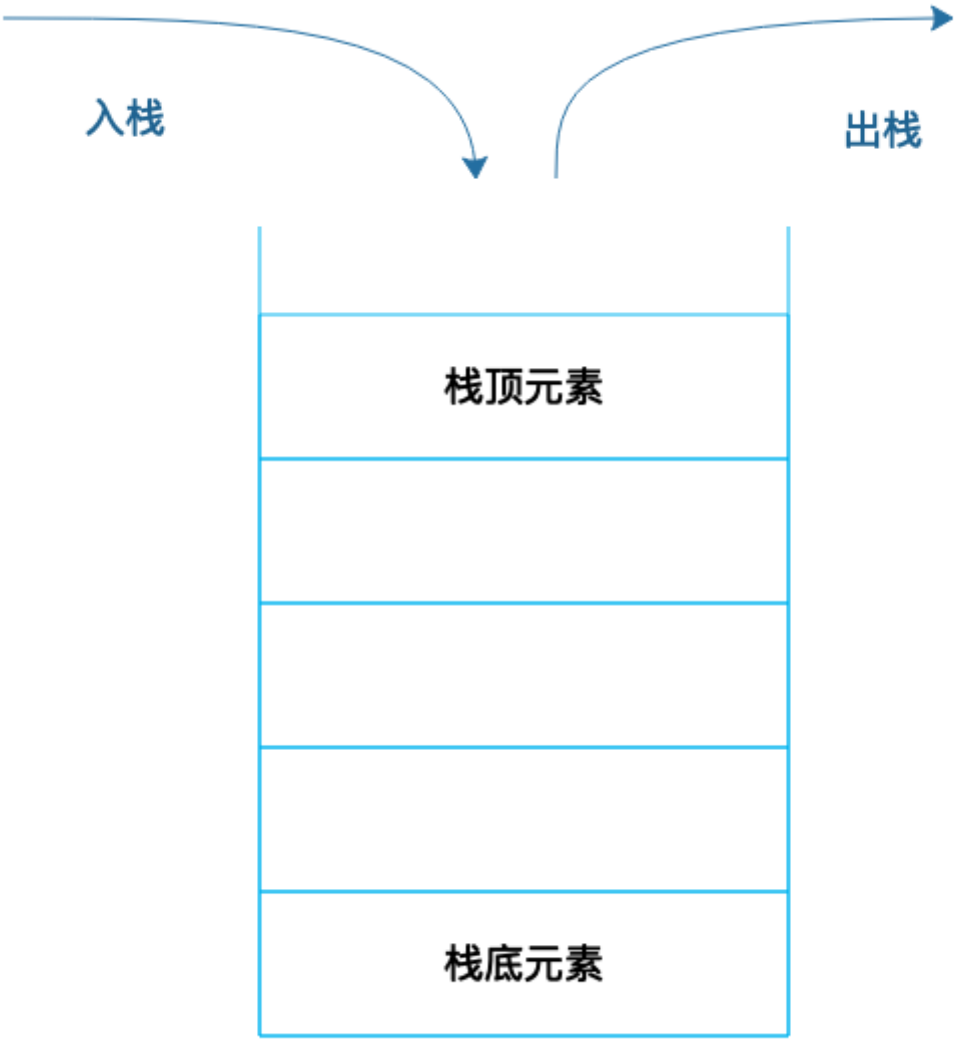


因为有 Hash 冲突的存在，所以“Hash 表的时间复杂度为什么是 O(1)？”这句话并不严谨，极端情况下，如果所有 Key 的数组下标都冲突，那么 Hash 表就退化为一条链表，查询的时间复杂度是 O(N)。但是作为一个面试题，“Hash 表的时间复杂度为什么是 O(1)”是没有问题的。

栈

数组和链表都被称为线性表，因为里面的数据是按照线性组织存放的，每个数据元素的前面只能有一个（前驱）数据元素，后面也只能有一个（后继）数据元素，所以称为线性表。但是对数组和链表的操作可以是随机的，可以对其上任何元素进行操作，如果对操作方式加以限制，就形成了新的数据结构。

栈就是在线性表的基础上加了这样的操作限制条件：后面添加的数据，在删除的时候必须先删除，即通常所说的“后进先出”。我们可以把栈可以想象成一个大桶，往桶里面放食物，一层一层放进去，如果要吃的时候，必须从最上面一层吃，吃了几层后，再往里放食物，还是从当前的最上面一层放起。



栈在线性表的基础上增加了操作限制，具体实现的时候，因为栈不需要随机访问、也不需要在中间添加、删除数据，所以可以用数组实现，也可以用链表实现。那么在顺序表的基础上增加操作限制有什么好处呢？

我们上篇提到的程序运行过程中，方法的调用需要用栈来管理每个方法的工作区，这样，不管方法如何嵌套调用，栈顶元素始终是当前正在执行的方法的工作区。这样，事情就简单了。而简单，正是我们做软件开发应该努力追求的一个目标。

队列

队列也是一种操作受限的线性表，栈是后进先出，而队列是先进先出。

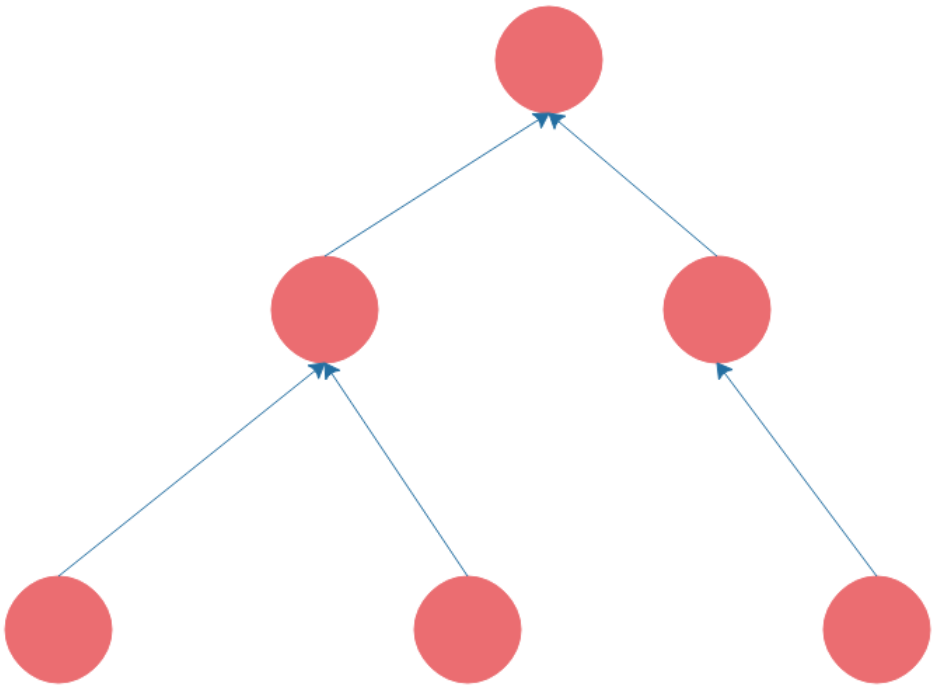
在软件运行期，经常会遇到资源不足的情况：提交任务请求线程池执行，但是线程已经用完了，任务需要放入队列，先进先出排队执行；线程在运行中需要访问数据库，数据库连接有限，已经用完了，线程进入阻塞队列，当有数据库连接释放的时候，从阻塞队列头部唤醒一个线程，出队列获得连接访问数据库。

我在上面讲堆栈的时候，举了一个大桶放食物的例子，事实上，如果用这种方式存放食物，有可能最底下食物永远都吃不到，最后过期了。

现实中也是如此，超市在货架上摆放食品的时候，其实是按照队列摆放的，而不是堆栈摆放的。工作人员在上架新食品的时候，总是把新食品摆在后面，使食品成为一个队列，以便让以前上架的食品被尽快卖出。

树

数组、链表、栈、队列都是线性表，也就是每个数据元素都只有一个前驱，一个后继。而树则是非线性表，树是这样的。



软件开发中，也有很多地方用到树，比如我们要开发一个 OA 系统，部门的组织结构就是一棵树；我们编写的程序在编译的时候，第一步就是将程序代码生成抽象语法树。传统上树的遍历使用递归的方式，而我个人更喜欢用设计模式中的组合模式进行树的遍历，具体我将会在设计模式部分详细讨论。

小结

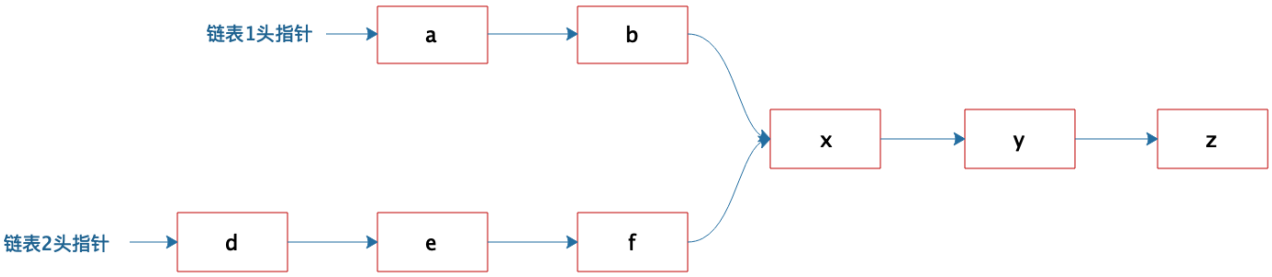
这是一篇关于数据结构的专栏文章，面试中问数据结构是一个非常有意思的话题，很多拥有绚丽简历和多年工作经验的候选人在数据结构的问题上翻了船，这些人有时候会解释说，这些知识都是大学时学过的，工作这些年用不着，记不太清楚了。

事实上，我很难相信，如果这些基本数据结构没有掌握好，如何能开发好一个稍微复杂一点的程序。但欣慰的是，在这些年的面试过程中，我发现候选者中能够正确回答基本数据结构问题的比例越来越高了，我也越来越坚定用数据结构问题当做是否跨过专业工程师门槛的试金石。作为一个专业软件工程师，不管有多少年经验，说不清楚基础数据结构的工作原理是不能接受的。

思考题

链表结构虽然简单，但是各种组合变换操作却可以很复杂。关于链表的操作也是面试官最喜欢问的数据结构问题之一，我在面试过程中喜欢问的一个链表问题是：

有两个单向链表，这两个单向链表有可能在某个元素合并，如下图所示的这样，也可能不合并。现在给定两个链表的头指针，如何快速地判断这两个链表是否合并？如果合并，找到合并的元素，也就是图中的 x 元素。



关于这道题，你的答案是什么呢？

欢迎你在评论区写下你的思考，我会和你一起交流，也欢迎把这篇文章分享给你朋友或者同事，一起交流一下。

上一篇　01 | 程序运行原理：程序是如何运行又是如何崩溃的?

下一篇　03 | Java虚拟机原理：JVM为什么被称为机器（machine）？

精选留言 (26)

写留言

funyoo
2019-11-20

说说我的思考：
方法1：在遍历链表1，看该结点是否在2中存在，若存在，即为合并。
遍历的方法：①两个for嵌套 ②根据1建立hash表，遍历2时在1中查找

方法2：计算两个链表的长度，谁长谁先“往前走”，待长链表未查看长度等于短链表是...
展开

作者回复: √
方法2 👍

7　18

大漠沙如雪
2019-11-20

思路特别清晰，但是都是点到为止呀，感觉差点啥，哈哈，可能后面的文章会具体介绍，更加详细。

3

乘坐Tornado的线程魔...
2019-11-20

思考题：第一步：分别遍历针对两个链表A、B（先不考虑长度差）。第二步：如果A链表先走到了末尾那么把A链表的指针指向B链表的第一个节点，继续遍历B链表；B链表的原始指针继续做遍历不变，走到末尾后，将B链表的指针指向A链表的第一个节点，继续遍历A链表。第三步：两个链表继续做遍历，如果分别指向的节点数值（地址）相同，那么这两个链表就是合并链表，这个节点就为合并的第一个节点。...
展开

1　3

breezeQian
2019-11-20

步骤一：将长度短的链表的元素构造成哈希表，key是指针，value是节点值。
步骤二：遍历较长链表找出合并的节点

展开

作者回复: √

3

扬帆丶启航
2019-11-20

计算两个链表的长度，用长的链表减去短的链表，计算出差值，长的链表先向后移动差值的个数，然后两个链表再同时移动，判断一下个节点的地址是否相同。

展开

1 2

小烽
2019-11-22

我的思考去下：能想到的就是遍历，结合前面几位的方法，做个比较。假设长度分别为m和n，m大。
方法一：先去遍历m 链表到m-n,然后同时遍历两个链表，比较大小，时间复杂度是O(m)

方法二：取一条链表的值放入hash 表，另一个链表进行遍历，在没有hash 冲突的情况...

展开

1

Geek_60eace
2019-11-20

获取2个列表的所有指针，进行交集运算，抛出第一个指针就是开始合并的地方

1

晶晶
2019-11-20

是我听过的课程里面觉得最让我思路清晰的课程~所以先打卡沙发一下☺

作者回复:

1 1

yusuf

2019-11-22

伪代码:
hashMap[char]bool map;
p1=l1;
p2=l2;
while (p1 != NULL || p2 != NULL) {...

展开

恐惧决定边界
2019-11-21

建立一个hash表，同时遍历两个链表，在不断往后遍历的过程中，与hash表中元素对比，存在则是交点元素，不存在就将元素添加进表中并继续遍历。

展开

黄海峰
2019-11-21

有点难想象，阿里大厂有面试官觉得这个问题难。。。

展开

木风
2019-11-21

把两链表值放入数组，从最末端往前找

展开

y欧尼酱
2019-11-20

遍历找出相同的节点。

展开

小太白52度
2019-11-20

1.将两个单链表顺序颠倒，即由尾指向头；
2.将两个新链表由头开始一一比较，直到两个元素不想等；
3.若第一元素即不等，则两个链表不合并，否则最后一个相等元素即为合并处的元素。

展开

握不住手中沙
2019-11-20

思考题：
方法一：遍历其中一个链表将链表元素放入Hash表中，然后再遍历另一个链表的元素是否在Hash表中出现；
方法二：比较两个链表的长度Len1，Len2，然后用双指针的方式，首先从较长链表的头部先走（Len1-Len2）步，然后两个指针同步向后查找，若两个指针指向相等则说明两个链...
展开

郭刚
2019-11-20

类似Oracle中的hash join的算法
展开

作者回复: √

远心
2019-11-20

实现了同时遍历两个单向链表的方法，欢迎拍砖：https://github.com/Huan-Rong/geektime-backend-technology-basics/blob/master/sources/2-the-time-complexity-of-hash-table/src/main/java/site/blbc/MergeDetect.java
展开

作者回复: 思路很赞

你这里利用了LinkedList的特性，按照题意，应该无法直接得到两个链表的长度。

幸福来敲门
2019-11-20

由于每个整型数据占据 4 个字节的内存空间，因此整个数组的内存空间地址是 0x1000 ~ 0x1039，根据这个，我们就可以轻易算出数组中每个数据的内存下标地址。利用这个特性，我们只要知道了数组下标，也就是数据在数组中的位置，比如下标 2，就可以计算得

到这个数据在内存中的位置 0x1008。

…

展开

作者回复: 谢谢指正，应该用十进制

2

小祺

2019-11-20

0x1000～0x1036写成了0x1000～0x1039

展开

BrokenSea1023

2019-11-20

虽然目前能力不足以让我给出方法，但是看到评论里的这些思路还是对我产生了极大的帮助！