搜索

To find the stuff that dreams are made of

引言

- 计算机最重要的功能之一就是在浩瀚的数据中找到用户所需要的信息
- 逐一查看全部的数据很容易找到你想要的东西，但是计算机的速度还没有快到能瞬间完成这一过程的程度
- 而且需要计算机进行查找的数据集往往是异常庞大的

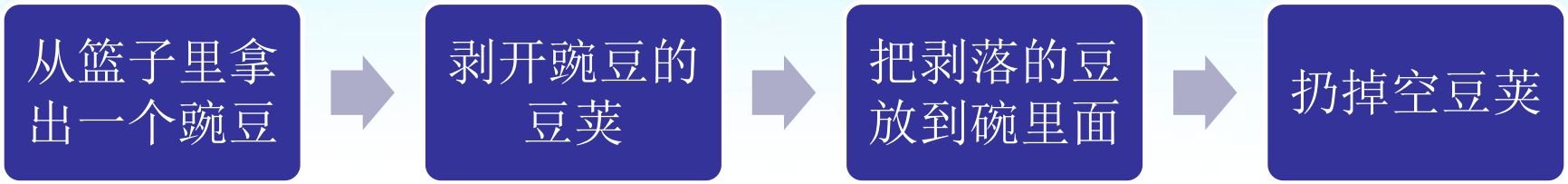
概要

- 这一章将向大家展示三种不同的搜索方式
  - 线性搜索
  - 二分搜索
  - 哈希（Hash）搜索
- 并通过对搜索方法的了解，对计算机科学的核心主题——算法有初步的认识

什么是算法

- 非正式的来说，**算法**（algorithm）是一系列的步骤，它规定如何完成一项任务
- 生活中的算法无处不在
  - 关于烹饪的算法（菜谱）
  - 在陌生城市准确定位的算法（道路指南）
  - 使用洗衣机的算法（标示在洗衣机内盖上）
  - 演奏音乐的算法（乐谱）

剥豌豆壳的算法



算法的形式定义

- 算法是定义一个可终止过程的一组有序的、无歧义的、可执行的步骤的集合
  - 有序意味着算法的各个步骤必须有非常明确的、顺序执行的结构
  - 算法必须由可执行的步骤组成，这也被称为有效性（effective）
  - 无歧义性和可终止性会在后面关于计算理论的章节中再做讨论

小游戏

- 画图形

算法的表示

- 一个算法的表示需要使用某种形式的语言
- 对于人类来说，这可能是一种传统的自然语言，或者可能是一种图形语言
- 对于计算机来说，这可能是一种描述程序指令的机器语言
- 前者常常容易引起误解，后者的描述过于抽象和单调

伪代码

- 伪代码（pseudocode）是一种在算法开发过程中非正式地表达思想的符号系统
- 通常我们通过放松正式机器语言对于表达最终算法的那些规则要求来构造伪代码
- 在不限定于特定程序设计语言的情况下来描述算法和表示问题

伪代码

赋值语句	• Remaining ← Checking + Saving
条件语义	• if (条件) then (活动) • else (活动)
循环语义	• while (条件) do (活动)
过程模块	• procedure 名称

算法的重要性

- 即使对于相同的任务，不同的算法之间也可能有非常明显的效率差别，有些算法的效率明显高于其他算法
- 虽然现在计算机本身的计算速度已经十分可观，但是有时候它们需要完成的任务规模非常庞大，处理时间可能会变得很长
- 这种情况下，设计快速有效的算法就显得尤为重要

算法的重要性

- 一些需要提高算法效率的例子
  - 在万维网中寻找包含有你名字或其他特定内容的网页
  - 在最节省燃料的前提下，寻找货车投递包裹的最佳路径
  - 为学校设计合理的时间表，保证在一天之内所有的班级都有课程安排

搜索问题

- 搜索（search）即是在一长串数列中查找某个特定数字的问题
- 我们这里以数字为例，而计算机中的所有对象都可以用数字来表达，因此同样的算法也能用于搜索任何类型的数据
- 我们将计算机需要搜索的数据，比如文字、条形码或者作者名字，称之为搜索关键词（search key）

搜索算法

- 对于存储了大量数据的计算机而言，由于需要在这些数据中快速读出信息，搜索功能是相当重要的
- 世界上最庞大的搜索问题是在网络世界中搜索信息，搜索引擎能实现同时为大量在线用户在数以亿计的网页中进行高速查询
- 能完成这样令人惊叹的工作，是因为他们使用了正确的算法

线性搜索

- 我们将通过“战舰游戏”来学习如何进行搜索
- 你和你的对手每人有相等数量的若干张卡片，每张卡片有随机的4位数字
- 每个人从手中的卡片中挑选一张作为自己的“战舰”，告诉对方数字，然后将卡片打乱顺序摊在面前
- 每人依次从对方那里选一张卡片翻开，首先找到对方战舰的人算作胜利

实践与思考

- 平均猜测几次才能找到“战舰”？
- 最多需要猜测几次？
- 如果对方作弊，“战舰”并不在卡片当中，什么时候你才会发现？
- 如果有100张卡片，需要猜测几次？
- 如果有10000或者更多的卡片呢？

线性搜索

- 在进行搜索时，我们从储存数据的开头开始找，直到找到制定数据时结束查找
- 这样方式被称为线性搜索（linear searching）也就是我们在战舰游戏中用到的方法
- 这种方法十分简单，所以经常用于数据规模较小的情形
- 但在数据规模增大的情况下，这种算法即使对于高速计算机来说也是非常慢的

线性搜索

- 假如一家超市货架上有10000种不同的货品，当收银员扫描一件货物的条形码时，计算机需要在10000种不同记录中去寻找这件商品的名称和价格
- 即使它能一秒内扫描1000次，查询全部的货物也要耗费10秒钟
- 这时客人也许已经等得不耐烦了，下次就会去别家超市购物了

线性搜索的伪代码

procedure Search(List, Target)

选择List中的第一项作为TestEntry

while (Target 不等于 TestEntry 而且
List中还有表项未检查)

do (选择List中的下一项作为TestEntry)

if (Target 等于 TestEntry)

then (宣布查找成功)

else (宣布查找失败)

二分搜索

- 同样以我们的“战舰游戏”为例
- 把手上的卡片从小到大排列后放在一行中，最小的数字放在最左边，最大的数字放在最右边
- 从正中间的位置选一张卡片，并进一步判断对方的“战舰”位于它的左边还是右边

思考这是如何实现的？

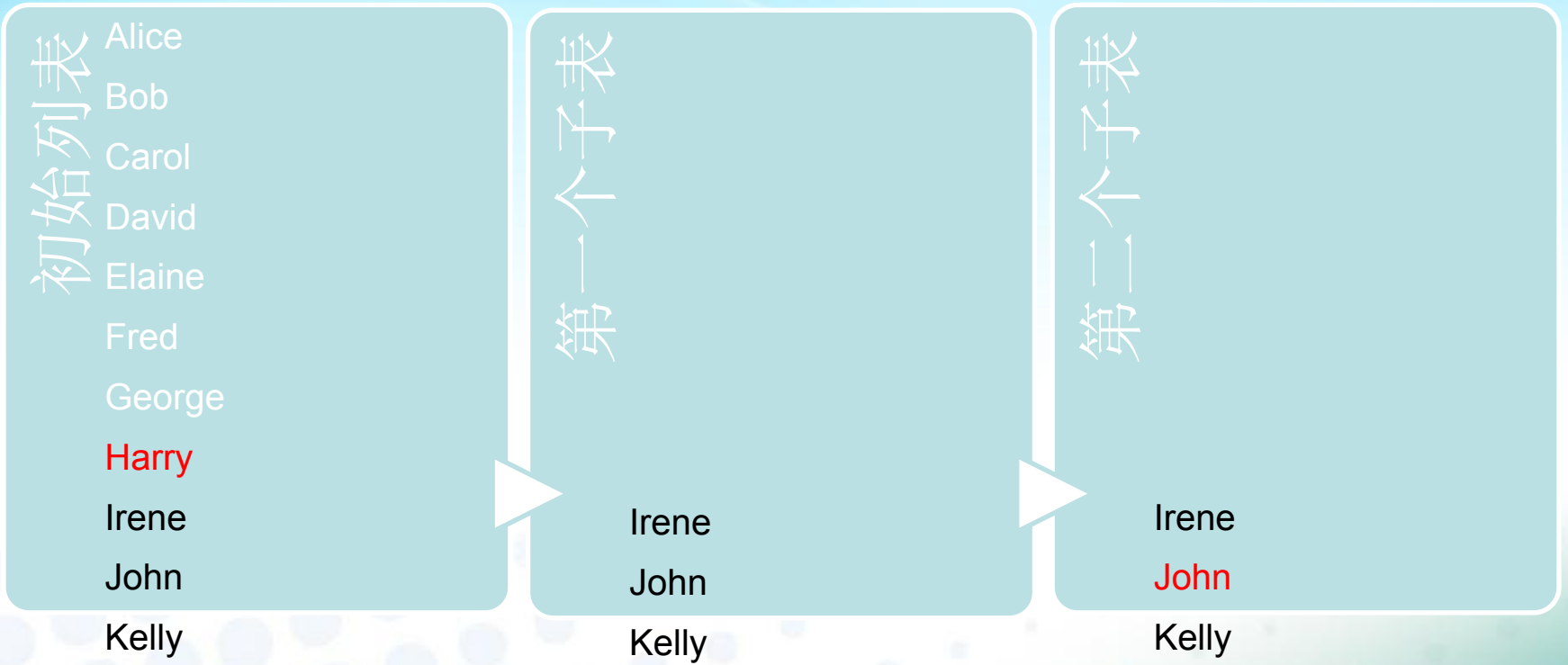
二分搜索

- 如果对方战舰的数字比你选出的卡片大的话，那么就说明“战舰”在右边数字较大的队列的这一半
- 如果对方战舰的数字比你选出的卡片小的话，那么就说明“战舰”在左边数字较小的队列的这一半
- 重复这个过程，你便能逐渐缩小“战舰”所在的范围，并最终找到它

实践与思考

- 平均猜测几次才能找到“战舰”？
- 最多需要猜测几次？
- 如果对方作弊，“战舰”并不在卡片当中，什么时候你才会发现？
- 如果有100张卡片，需要猜测几次？
- 如果有10000或者更多的卡片呢？

二分搜索John的例子



二分搜索

- 使用二分搜索法（binary search）花费的猜测次数比使用线性搜索法少
- 只用检查队列的中间项就可确定搜索关键词位于哪一半队列
- 这样一来，每猜一次相当于将待查找的目标数量减少了一半

二分搜索

- 再回头看看超市的例子
- 如果采用二分搜索方法在10,000件货品中查找，现在仅需用到14次搜索
- 这几乎可以在百分之一秒内完成，快到让人难以察觉
- 如果采用二分搜索法，在1,000,000 个货品中需要搜索多少次才能找到目标？

二分搜索的伪代码

```
procedure Search(List, Target)
if (List为空)
        then (报告查找失败)
else
        选择List中的中间项作为TestEntry
        case 1: Target = TestEntry
                (报告查找成功)
        case 2: Target < TestEntry
                (在List中位于TestEntry项之前的部分查找
                Target，并报告查找结果)
        case 3: Target > TestEntry
                (在List中位于TestEntry项之前的部分查找
                Target，并报告查找结果)
```

二分搜索

- 思考：如果在前面的表项中查找以下的名字，伪代码所表述的算法分别会有怎样的执行过程
  - John
  - Alice
  - Elaine
  - Oliver
  - Smith

哈希搜索

- 这是一种更高效的搜索算法
- 与二分搜索相比，它无需对手中的卡片进行排列，而是将卡片的数字进行哈希化（hashing），从而为每张卡片根据其搜索关键词指定一个类别
- 这样一来，当你要查找该关键词时，便能精确定位需要查找的类别
- 哈希算法是计算机中搜索数据的最快方法

哈希函数

- 一个简单的哈希函数是将卡片数字的每个数位相加，取其总和的末尾数字。
- 总和的最后一位数字决定了将其分配到哪一个类别，因此一共有10类（0到9）
- 比如，编号为2345的卡片，将各位数字相加2+3+4+5，得到14。由于总和的末位为4，所以这张卡片将被分配到数字4的类别当中

哈希搜索

- 再回到战舰游戏当中
- 准备10个类别的标签（0到9），算出每张卡片的哈希值，并将它们归放到相应的类别当中去
- 最后再进行“战舰”游戏，比如你的目标“战舰”是5678，各数位相加之后得到26，说明你只要在对方标为6的类中寻找目标即可

实践与思考

- 平均猜测几次才能找到“战舰”？
- 如果对方作弊，“战舰”并不在卡片当中，什么时候你才会发现？
- 如果你想要让你的“战舰”不那么容易被找到，你需要把它藏在哪一类里面？
- 如果你想要尽快找到对手的“战舰”，你最希望它被归到哪一类里面？

哈希搜索

- 通常来说，哈希搜索是最快的搜索算法
- 但是它的运行速度取决于类别中对象的数量和类别的数量
- 在哈希函数的设计过程中，我们通常希望每个类别包含尽量少的对象
- 这样也就意味着计算机通过简单的计算之后就能快速锁定关键词储存的位置

哈希函数

- 哈希函数的设计有许多类型
  - 除留余数法
  - 数字分析法
  - 平方取中法
  - 折叠法
- 我们前面在“战舰游戏”中采用的哈希函数属于折叠法

哪一种速度最快？

哪一种需要每个对象采用特定的方式排列在一起？

它们分别适用于什么情况？

思考：本章中介绍的三种搜索方法各自的优点和缺点分别是什么

相关的进阶讨论

二分搜索

- 采用二分搜索法搜索关键词的速度很快，但如果你想要增加一个关键词的话，搜索速度就会明显变慢
- 因为新增对象首先还需要被添加到关键词序列中，二分搜索只能在排好序的关键词序列中应用，而在尾部新增一个关键词会打乱原有的序列顺序
- 如果需要存在插入、删除、修改的情况，一般使用“二叉搜索树”的方法

哈希搜索

- 哈希搜索法中使用的“类别”一般被称之为“桶”（bucket）
- 在我们前面的例子中，锁定一个桶后，在桶中采用的是线性搜索
- 通常桶内包含的对象数量不多，这一点不会对运行速度产生太大的影响
- 一旦桶内包含的对象数量较多的话，我们就需要对每个桶内的数据再进行处理了——在搜索中搜索

哈希搜索

- 事实上，在哈希过程中，映射到同一个桶中的现象是常常发生的
- 考虑一个40人的班级，其中有两人生日相同的概率是多少？
- 事实上，这样概率高达90%
- 而一个拥有57名学生的班级中，两人拥有一样生日的概率则高达99%

哈希搜索

- 同样的情况也存在哈希过程中
- 在之前的“战舰游戏”中，即使我们使用多于10个的类别，不同的卡片被归于同一类别的概率仍然很大，这种现象我们称为“冲突”（collision）
- 这也就是为什么对计算机来说，有效地处理哈希冲突是至关重要的

参考资料

- 更多关于二分搜索和哈希搜索的内容可以参考以下资料
  - 《数据结构——用面向对象方法与C++语言描述》的5.5节与6.5节
  - 《算法导论》的第11章与第12章

Thank you for attending

Q&A