排序

Let all things be done
decently and in order

引言

- 几乎所有计算机中的序列都是排过序的
  - 电子邮件列表按照日期排序，最新的邮件被放置在最顶端
  - 音乐播放器中的歌曲按歌曲名或者歌手名排列在一起，以便你能快速找到喜欢的那首
  - 文件名则往往是按字母顺序排列的
- 那么计算机是如何进行排序的呢？

概要

- 这一章将向大家介绍用于排序的三种简单算法
  - 选择排序
  - 插入排序
  - 冒泡排序
- 同时通过了解它们工作的原理，找出他们的局限与不足

排序的意义

- 对序列进行排序有助于更快地找到我们想要的东西，以及更好的实现我们希望完成的任务
- 例如，在我们前面学习过的二分搜索法，就必须在查找对象已经处于排好序的状态下才能顺利执行
- 在许多问题当中，排序都是首先需要完成的步骤

排序的限制

- 在计算机中进行排序，与我们平时排列对象相比，是有一定的限制的
- 简单的说，我们一般要求在序列的内部将对象排序，而不能把序列移动到其它位置
- 通常，序列储存在一片连续的空间中，就像在书立当中依次紧密排列的图书
- 这种限制是很典型的，因为这样能更有效的利用存储空间，同时也更方便我们在排序之后进行其它操作

排序的限制

- 另外，计算机每次只能对比两个数据，而人的习惯于能够一次性比较多个数据
- 如果你让计算机在序列“10，50，2，30，8”中找到最小数，计算机并不能够一下便“看”到“2”是最小的数字，它只能一次对两个数字进行比较
- 接下来的例子当中，我们都是基于这样的限制，因此只能一次比较两个数据

排序的形式

- 假设你需要排列的对象是一组外观一致的重物，你只能使用天平来比较两个不同物体的重量，且每次只能比较两个被测物
- 或者考虑你有一组标有数字的卡片，但要比较其中两张大小时，只能让你的同伴查看上面的数字，然后告诉你比较的结果
- 排序是将一组无序关键字（key）变成一组有序输出的过程
- 上面两个例子中，什么是关键字？

选择排序

- 首先来考虑一个更为简单的问题：要找出重量最轻的物体（或最小数字的卡片），应当使用怎样的方法能最快得出结果呢？
- 最佳方案是让最轻的重物始终都在天平的同一端，然后拿其它重物过来和它进行比较，如果遇到更轻的重物，就用这个更轻的取代之前最轻重物的位置

实践与思考

- 一共比较了多少次才找到最轻重物
- 如果要在100个被测物中找到最轻的一个，一共需要比较多少次？
- 能找出重物数量 n 与找到最轻被测物所需要进行比较的次数 c 之间的关系么？
- 用两个重物和一个重物的例子来测试一下你的公式是否正确

排序中的比较次数

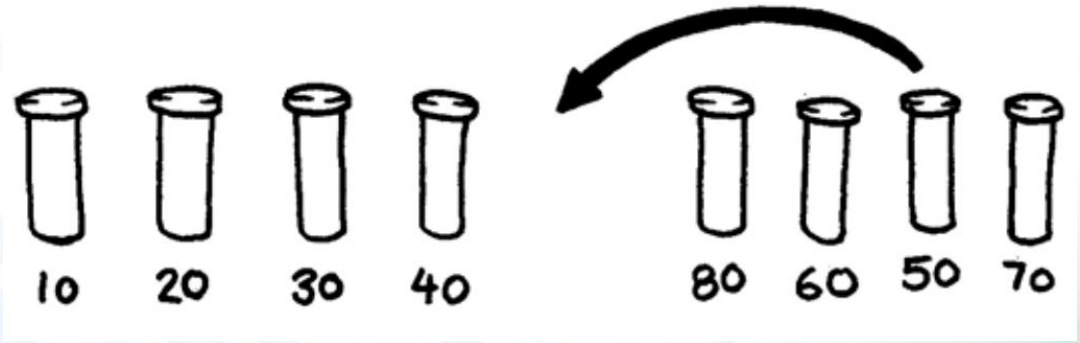
- 了解排序算法需要执行比较的次数是相当重要的，它关系着算法在计算机中执行的效率，而需要进行比较的次数与算法耗费的时间几乎成正比
- 比起测量算法耗费的时间，记录需要比较的次数更方便，因为耗费的时间可能因排序的操作者或比较工具的不同而有差异，而比较次数在任何人使用任何工具的情况下都是相同的

选择排序

- 通过刚才例子中找到最小重量的方法，我们可以将一整个序列排序
- 先选一个最轻的重物放在一边，然后在剩下的重物中挑选最轻的那个，依此类推
- 重复上述步骤直到所有的重物都被挑出来
- 完成后它们即已从轻到重被整齐排列
- 这个方法被称为选择排序（selection sort），因为你总是持续地选择最小的数值

选择排序

- 比如下图中，重物10、20、30和40已经被选出并放在一旁，下一步就是在剩下的4个重物中找出最轻的那个，即50。当把50选中并移动到左边那组后，就只剩下3个重物未被排序了



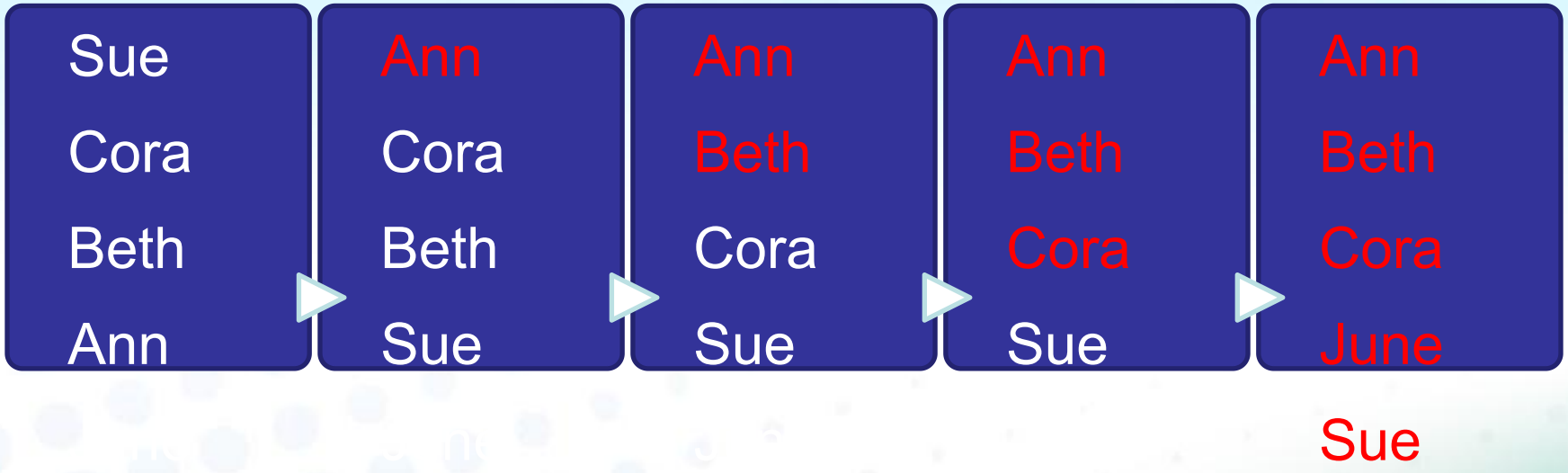
选择排序

- 随机排列8个重物，然后使用选择排序法将他们按重量排序。计算一下一共需要比较多少次
- 如果你采用的方法正确，那么找到第一个最轻重物需要比较7次，找到次轻的重物需要比较6次，依次类推。所以得到排序的比较结果总共需要比较的次数为 7 + 6 + 5 + 4 + 3 + 2 + 1

选择排序

- 如果你用错了排序方法，那么在对一个大序列排序时将耗时巨大，就算用一台高速计算机结果也一样
- 实际上，我们刚才使用的选择排序法并不是一个很好的方法
- 如果用该方法对1000个数据排序，我们将需要比较近50万次（准确说是499500次）

另一个选择排序的例子


Sue
Cora
Beth
Ann
Ann
Cora
Beth
Sue
Ann
Beth
Cora
Sue
Ann
Beth
Cora
Sue
Ann
Beth
Cora
June
Sue

选择排序的伪代码

procedure Sort(List)

i ← 1;

while (i的值不超过List的长度N) **do**

找出第i项开始到List末端最小的元素

将最小元素与第i项元素交换位置

I ← i + 1;

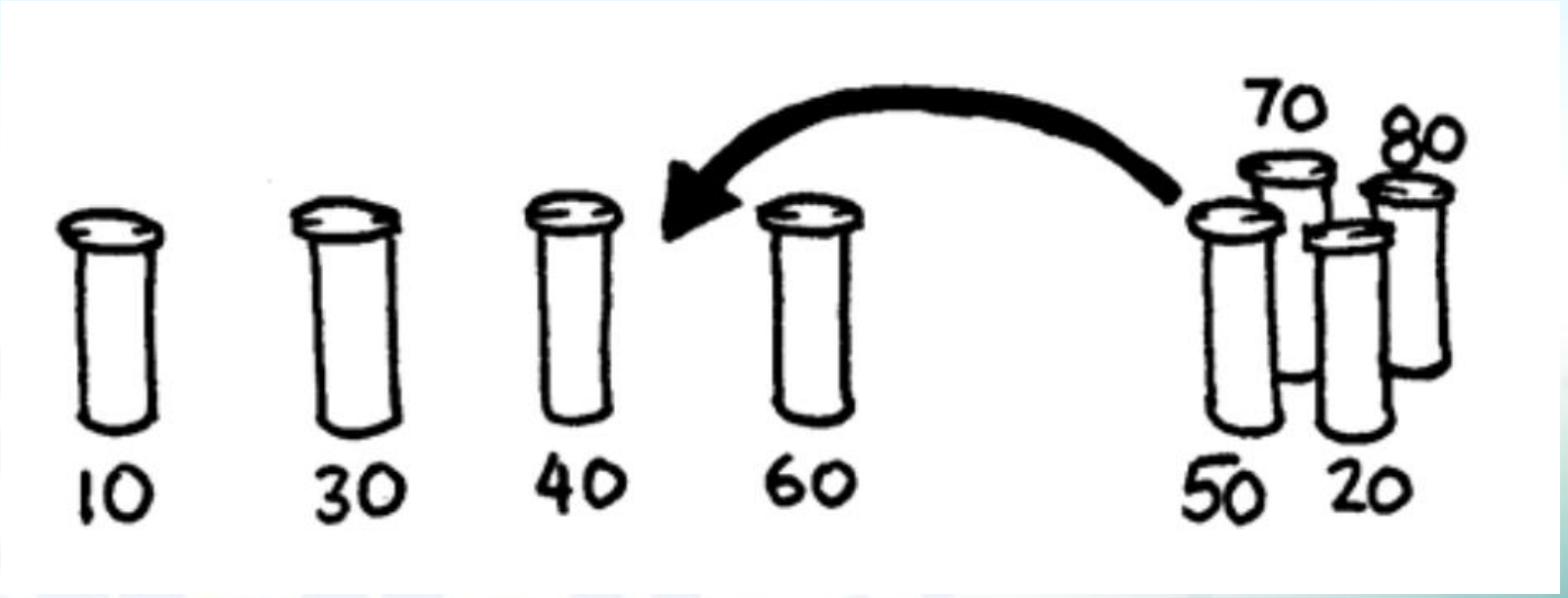
下一章我们会谈到其他更为高效的排序方法

但除了在某些特定条件下，这两种方法并不会比选择排序效率更好

解决排序的另外两种方法

插入排序

- 插入排序（insertion sort）的工作原理为：在一个未排序的序列中依次移出每个对象，将它们插入到有序序列中的正确位置。



插入排序

- 每成功插入一次，即意味着未排序的对象减少了，排好序的对象增加了，直到整个序列被完全排列好
- 想想这个过程跟我们生活当中的哪一项娱乐活动非常相似？

插入排序 vs 选择排序

- 共同点：都是将序列分成两部分——已排好序的和未排好序的
- 不同点
  - 选择排序是考虑应当从未排序的序列中，挑选哪一个数值加入到已排序序列
  - 插入排序的重点在于如何将数值放入已排序序列中的正确位置上

插入排序

- 让我们按照下列步骤来执行一次插入排序
- 第一步，随机挑出一个对象作为左侧有序序列的第一个物品
- 第二步，从右侧未排序的序列中选择第二个对象，用天平来决定它与第一个对象的大小
- 第三步，选择第三个对象，用天平来决定第三个对象同前一次比较中较大者之间孰大孰小

插入排序

- 如果第三个对象较大的话，将它放在左侧序列的最右端，如较小的话，将它与第一个对象比较，根据结果来决定将它放在第一个对象的左侧还是右侧
- 按上述方法持续从右侧的乱序序列中选择对象，并将其插入到左侧序列中正确的位置上，如果它比前一个被插入对象大的话，则将它直接插入到前一个被插入对象的右侧

实践与思考

- 假设有8个重物，采用插入排序法对它们进行排序，记录一共需要进行多少次比较
- 你觉得最多需要多少次比较？
- 最多的比较次数发生在怎样的情形下？
- 你觉得最少需要多少次比较？
- 最少的比较次数发生在怎样的情形下？

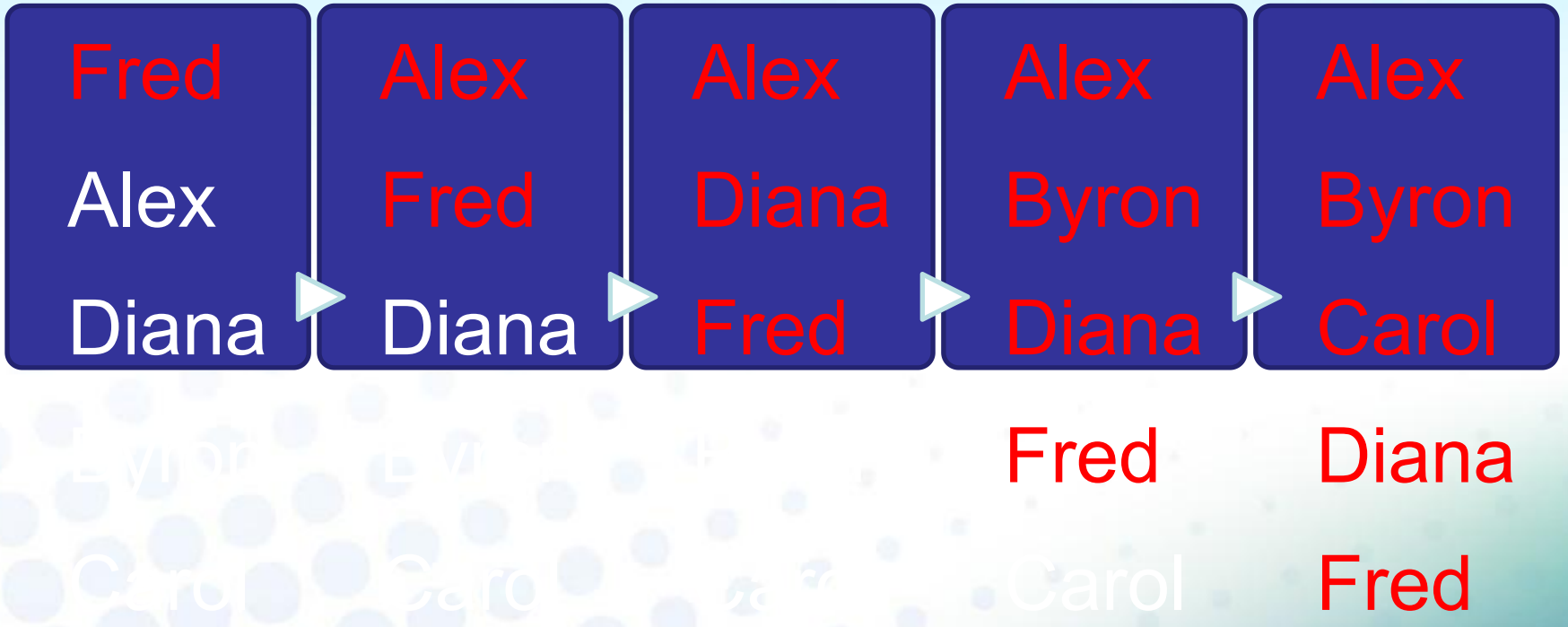
插入排序

- 现在你应该已经发现，插入排序要比选择排序快一点：平均下来你只用将每个重物与其他半数对象进行比较即可
- 当然这只是个平均值，实际情况你的比较次数有时会少一些，有时则会多一些
- 尽管如此，插入排序仍然是一个很慢的方法，例如采用该方法对1000个对象的序列进行排序平均需要比较25万次

插入排序

- 有没有能够进一步减少比较次数的方法？
- 二分搜索！
- 然而，这并不意味着算法的执行效率会有显著的提高，因为插入排序除了比较之外，还需要对序列进行移动（前面提到的对空间的限制），这会成为算法新的瓶颈
- 另外使用二分搜索会使我们失去只每轮插入比较一次的机会

插入排序的例子


Fred
Alex
Diana
Alex
Fred
Diana
Alex
Diana
Fred
Alex
Byron
Diana
Fred
Carol
Alex
Byron
Carol
Diana
Fred

具体步骤


Alex
Diana
Fred
Carol
Alex
Diana
Fred
Carol
Alex
Diana
Fred
Carol
Alex
Byron
Diana
Fred
Carol

插入排序的伪代码

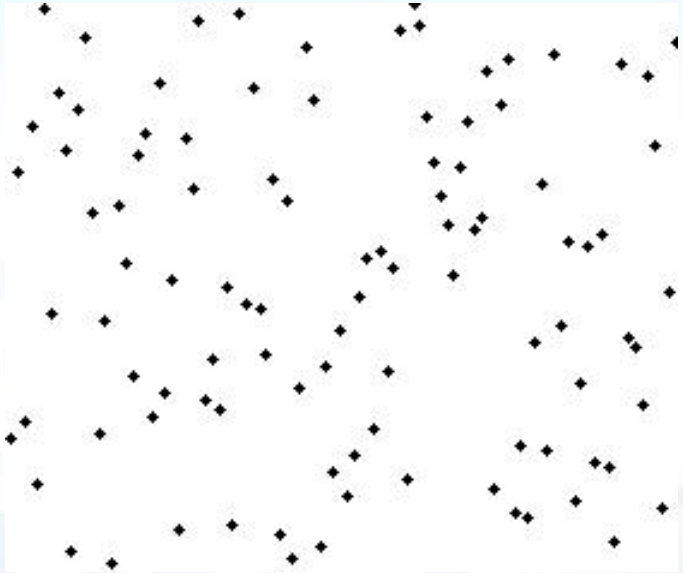
```
procedure Sort(List)
N ← 2;
while (N的值不超过List的长度) do
        (把List的第N项作为主元项          ;
        把主元项移到一个临时位置使List留出一个空位置;
        while (如果这个空位置上面存在一个数值并且那个
                数值比主元大) do
                (把这个项向下移动到空位置上使该项上面留
                出一个空位置)
        把主元项插入到List的空位置上
        N ← N + 1;
        )
```

冒泡排序

- 冒泡排序（bubble sort）是一种需要将整个序列反复扫描，并交换所有相对位置错误的相邻数据的方法
- 当检查整个序列发现整个序列不用交换任何数据时，便证明序列已经排好序了
- 这种方法的效率并不高
- 事实上，它是排序方法中最差的一个

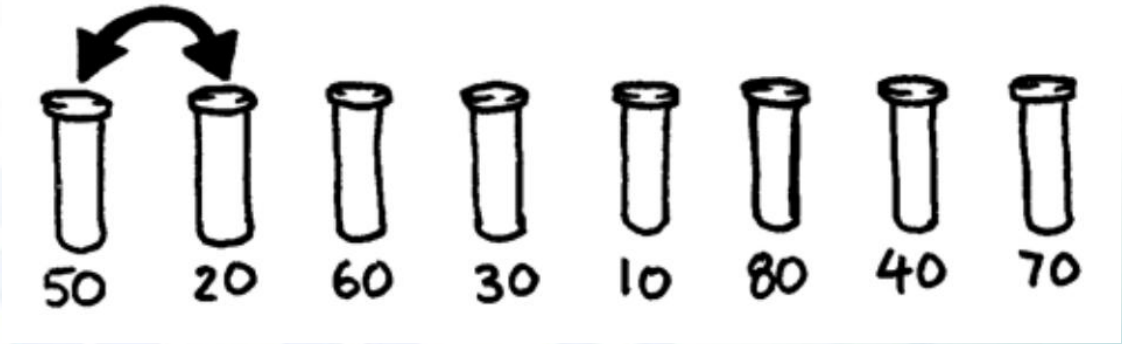
冒泡排序

- 冒泡排序是最容易被理解的一种排序方法，
- 它的得名来自于其方法的特性：总是小数往前放，大数往后放，与气泡上升的过程类似



冒泡排序

- 将8个重物随机排列，然后用冒泡排序法对它们排序
- 比较下图所示的第一组重物，如果它们的相对位置错误，则交换他们的位置
- 接着比较第二组重物（例子中不需要交换）



冒泡排序

- 如果一直检查到序列的最右边都不需要交换任何两相邻重物的时候，说明排序已经完成
- 否则需要将刚才的过程从头再来一次，直到整个序列都不需要交换任何数据
- 这就意味着整个序列已经按有序的形式排列了

实践与思考

- 假设有8个重物，采用冒泡排序法对它们进行排序，记录一共需要进行多少次比较
- 你觉得最多需要多少次比较？
- 最多的比较次数发生在怎样的情形下？
- 你觉得最少需要多少次比较？
- 最少的比较次数发生在怎样的情形下？

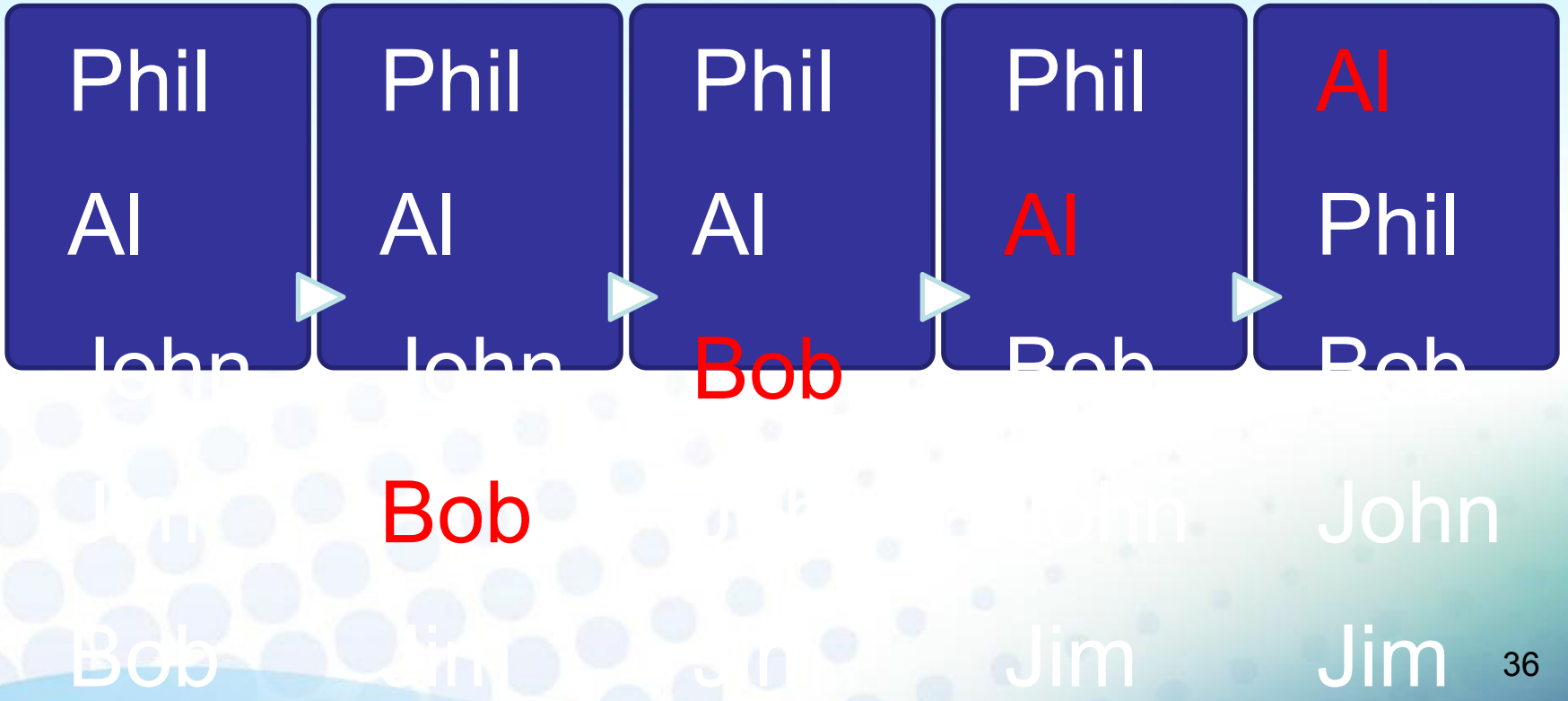
冒泡排序

- 在冒泡排序中，将整个序列检查过一遍后，你会发现最右边的对象已经位于正确的位置了，下一次检查只需要比较余下的7个对象，再下一次只用比较余下的6个对象，依次类推
- 这意味着冒泡排序最多只需要进行 7 + 6 + 5 + 4 + 3 + 2 + 1 = 28 次比较

冒泡排序

- 之所以说冒泡排序是效率最差的排序，是因为它拥有和选择排序相同的比较次数，但需要付出大量额外的交换操作
- 它的优点除了简易性之外，还在于其可以在序列已经排好序之后就停止执行，这是插入排序和选择排序所不能实现的

第一次迭代


Phil
Al
John
Bob
Phil
Al
John
Bob
Phil
Al
Bob
Phil
Al
Bob
Jim
Al
Phil
Bob
John
Jim

余下的迭代


Al
Phil
Bob
John
Al
Bob
Phil
Jim
Al
Bob
Jim
Phil
Al
Bob
Jim
John
Phil

冒泡排序的伪代码

```
procedure Sort(List)
N ← List的长度;
while (N>0而且交换还在发生) do
        (选择第一项为当前项Current;
        while (当前项Current未达到长度N) do
                if (当前项与相邻项逆序) then
                        (交换这两项并报告交换发生)
                选择下一项为当前项Current;
        N ← N - 1;
        )
```

哪种平均比较次数较少？

哪种移动次数和交换次数较少？

哪种能够快速发现排序已完成？

思考：比较三种排序方法的优点与缺点

进阶讨论

- 尽管三种排序算法需要的平均比较次数各不相同，但是它们在最坏情况下需要的次数均为 $1 + 2 + 3 + \ldots + (n - 1) = n^2/2 - n/2$
- 事实上，公式中包含的n^2非常关键：如果n变大两倍，那么意味着比较的次数增大到4倍；如果n变大10倍，那么意味着比较的次数增大到100倍

进阶讨论

- 正因为公式中前半部分的 值如此重要，因此三种算法通常被称为“n^2”排序法，或称为二次排序法
- 在下一次课程中，我们将学习其它不会被n^2影响的排序方法，对于大规模数据集合来说它们的排序速度将显著提高
- 这里我们已经讨论到了关于算法复杂度的内容，可参考《算法导论》第3章

Thank you

Q&A