从圆桌问题谈数据结构的综合运用

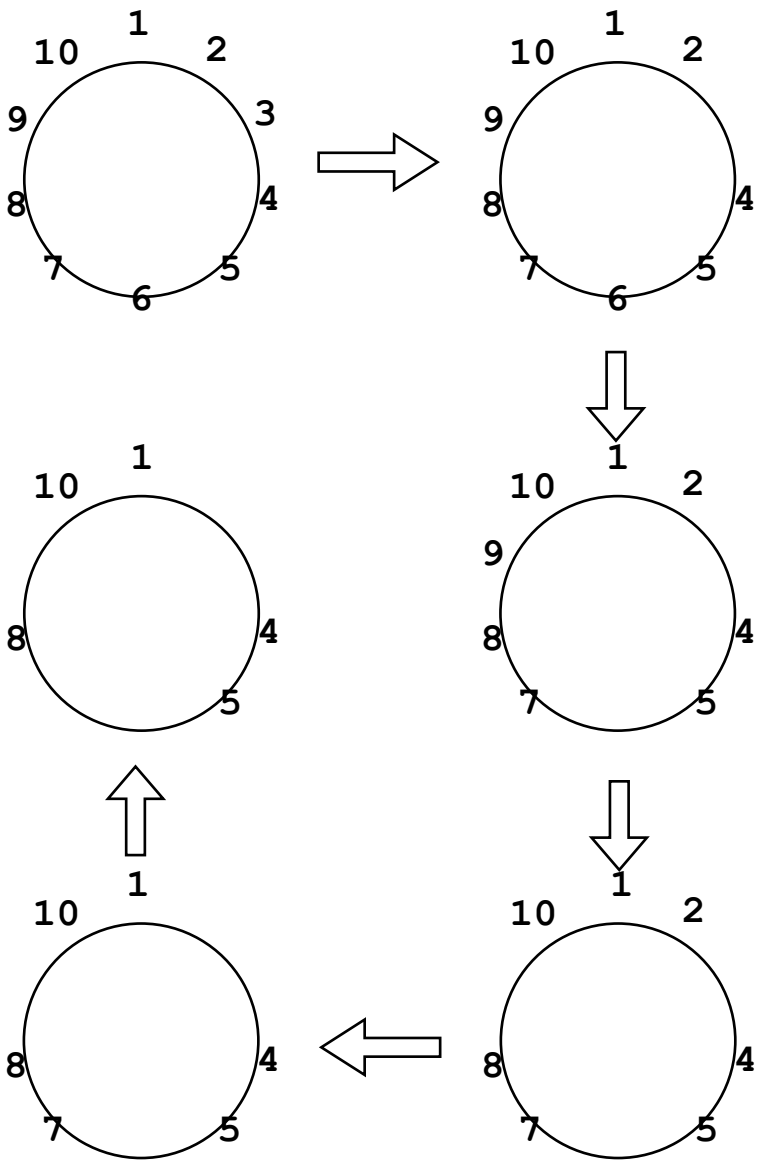
圆桌问题

题目：圆桌上围坐着 2n 个人。其中 n 个人是好人，另外 n 个人是坏人。如果从第一个人开始数数，数到第 m 个人，则立即处死该人；然后从被处死的人之后开始数数，再将数到的第 m 个人处死...依此方法不断处死围坐在圆桌上的人。试问预先应如何安排这些好人与坏人的座位，能使得在处死 n 个人之后，圆桌上围坐的剩余的 n 个人全是好人。

输入：文件中的每一行都有两个数，依次为 n 和 m，表示一个问题的描述信息， n≤32767，m≤32767。

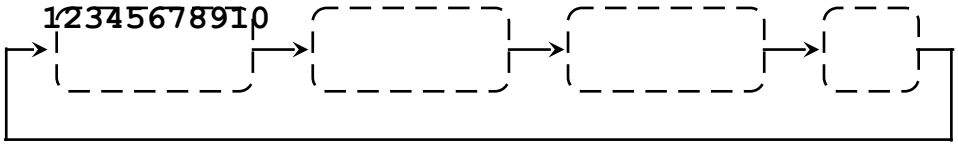
输出：依次输出每一个问题的解。每一个问题的解可以用连续的若干行字符来表示，每行的字符数量不超过 50。但是在一个问题的解中不允许出现空白字符和空行，相邻的两个问题的解之间用空行隔开。用大写字母 G 表示好人，大写字母 B 表示坏人。

圆桌问题实现思想图示（$n=5,m=3$）

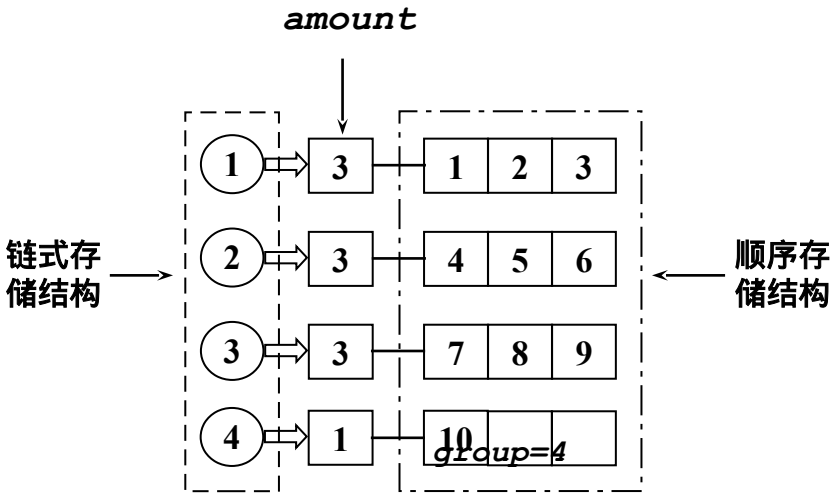


分段式数据结构示意

(思想模型)



(实际模型)



共进行 1+2+2+3+5=13 次操作

改进前后程序效率比较

（测试机器：*P166*）

测试数据	线性表 “查找”法	“优化直接定位”法	
		amount=400	改进前用时是改进后的多少倍
n=200 m=100	0.000s	0.000s	/
n=1000 m=50	0.440s	0.000s	/
n=32767 m=200	5.870s	0.930s	6.312
n=32767 m=1000	29.440s	0.980s	30.041
n=32767 m=10000	294.120s	1.260s	233.43
n=32767 m=20000	588.530s	1.590s	370.14
n=32767 m=32767	963.560s	1.970s	489.12

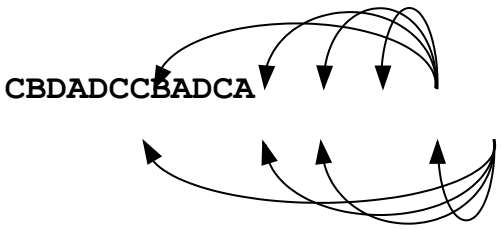
引申

- 横向延伸——约瑟夫环类的问题

  如：《翻牌游戏》、《猴子选大王》

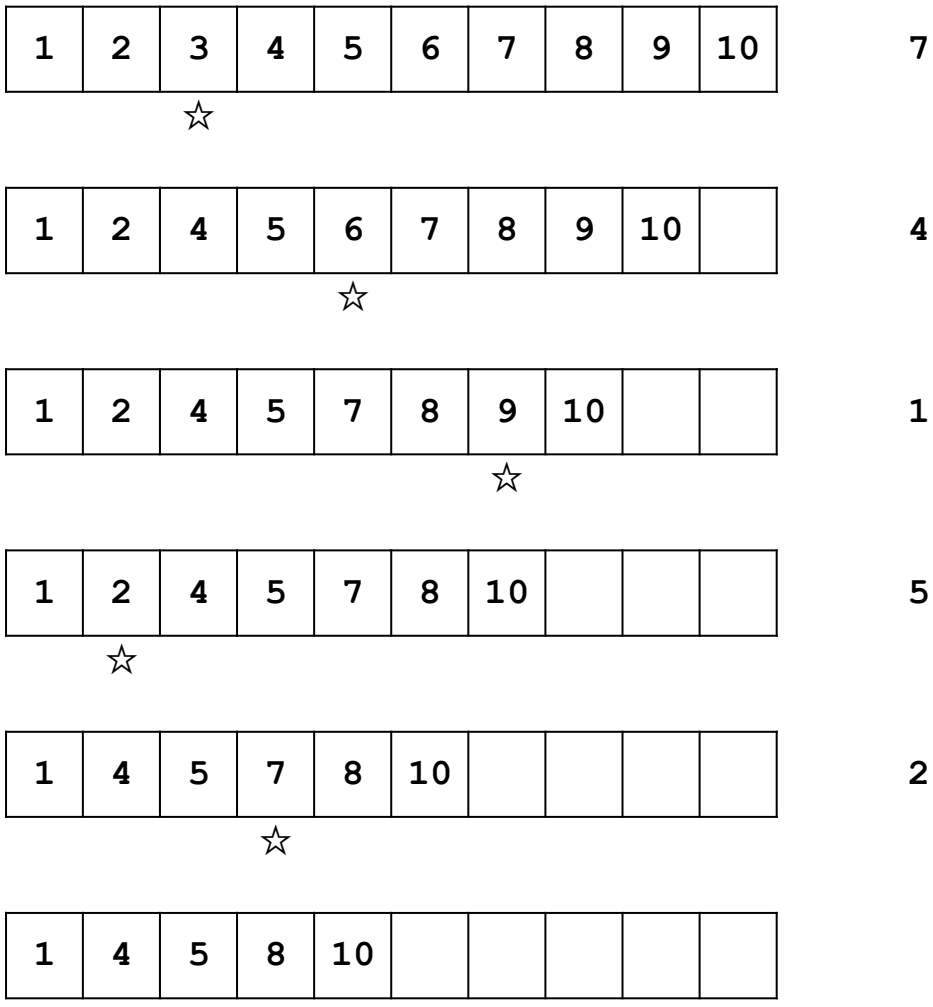
- 纵向延伸——数据结构的综合运用

  在解决一些数据规模较大的问题时有很好的效用。如《隐藏的码字》*(IOI'99)*。在解决这道题目时，如果建立起链式和顺序相结合的数据结构（如下图），程序效率就比较高。



链式和顺序相结合的数据结构实现简单，效果显著，应用比较广泛。当然还有其它的结合，比如二叉堆和顺序结构的一一映射（单射），在解决某些问题时会有很好的效果。

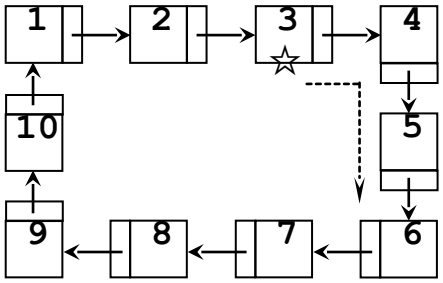
顺序存储结构操作示意



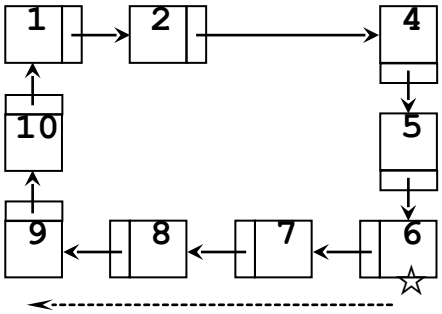
共进行 7+4+1+5+2=19 次操作，时间复杂度 $O(n^2)$。

链式存储结构操作示意

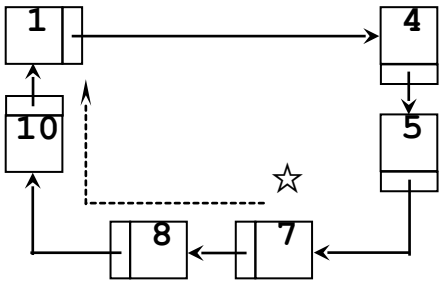
Step 1



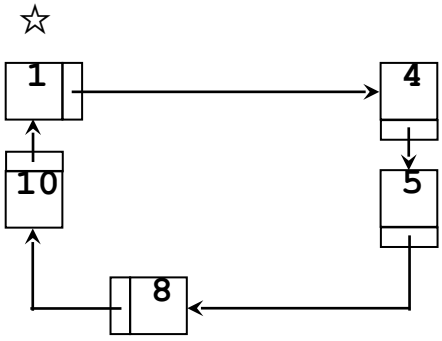
Step 2



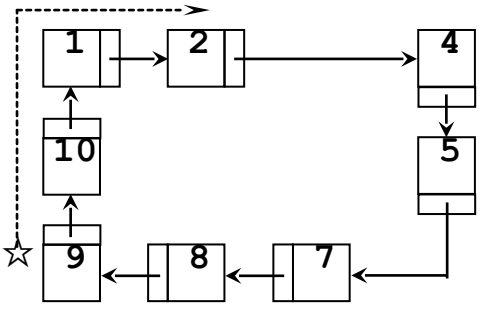
Step 3



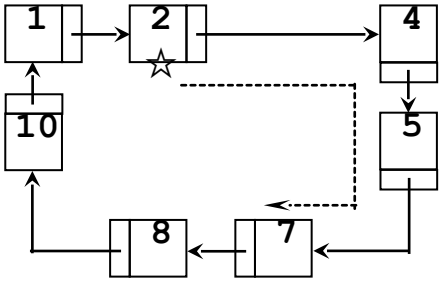
Step 4



Step 5



Step 6



共进行 5×3=15 次操作，时间复杂度 *O(nm)*。

从圆桌问题谈数据结构的综合运用

例．圆桌问题*(AH'99)*

题目： 圆桌上围坐着 $2n$ 个人。其中 n 个人是好人，另外 n 个人是坏人。如果从第一个人开始数数，数到第 m 个人，则立即处死该人；然后从被处死的人之后开始数数，再将数到的第 m 个人处死……依此方法不断处死围坐在圆桌上的人。试问预先应如何安排这些好人与坏人的座位，能使得在处死 n 个人之后，圆桌上围坐的剩余的 n 个人全是好人。

输入： 文件中的每一行都有两个数，依次为 n 和 m ，表示一个问题的描述信息。约束条件： $n \le 32767$ ， $m \le 32767$ 。

输出： 依次输出每一个问题的解。每一个问题的解可以用连续的若干行字符表示，每行字符数量不超过 50。但是在一个问题的解中不允许出现空白字符和空行，相邻的两个问题的解之间用空行隔开。用大写字母 G 表示好人，大写字母 B 表示坏人。

解法：

思想：模拟实际过程，寻找前 n 个被"处死"的人的位置。

1． 普通解法——线性表"查找"法

1 用顺序存储结构实现

用数组记录当前所有未被处死的人原来的位置，初始值为 1..2n。可根据前一个被处死的人在数组中的位置（即下标）直接定位，找到下一个应该被处死的人在数组中的位置，然后删去，并将它后面的元素全部前移一次。

2 用链式存储结构实现

用链表记录当前所有未被处死的人原来的位置，初始值为 1..2n。每处死一个人后，只要将这个结点直接从链表中删去即可，然后指针后移 (m-1) 次，找到下一个应该被处死的人。

2． 改进解法——"优化直接定位"法

总体思想就是在较好地实现"直接定位"的基础上，尽量避免大规模的元素移动。

设计出的数据结构如图 1 所示：其中 group 表示将原来的数据分为几段存储；每一段的开头记下的 amount 值表示此段中现有元素的个数。随程序的运行，amount 值是不断减小的。

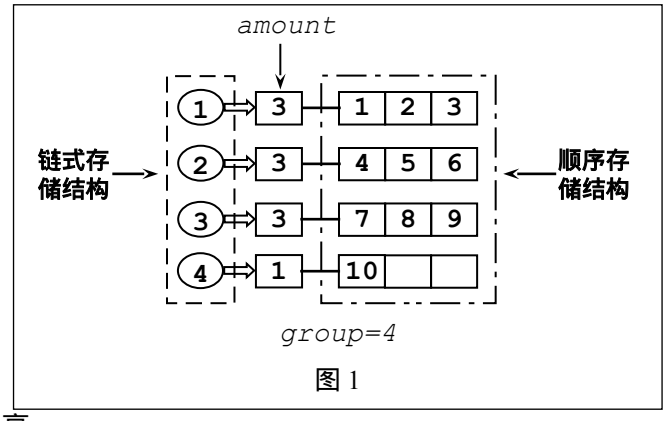


图 1

这种结构可以看作是**链式**存储结构和**顺序**存储结构的结合产物，兼具这两种存储结构的优点。运用了这种存储结构后，程序效率显著提高。

引申

- 横向延伸——其它约瑟夫环问题

  如：《翻牌游戏》、《猴子选大王》

- 纵向延伸——数据结构的综合运用

  在解决一些数据规模较大的题目时有很好的应用。如《隐藏的码字》*（IOI'99）*。在解决这道题目时，如果运用链式和顺序相结合的数据结构（如图 2 所示），程序效率就比较高。

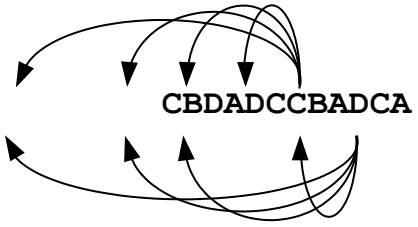


图 2

链式和顺序相结合的数据结构实现简单，效果显著，应用比较广泛。当然还有其它的结合方式，比如二叉堆和顺序结构的一一映射（单射），在解决某些问题时有非常好的效果。

小结

"网络式思维方式的核心是**联系**"。在做题目时，我们应深挖题目所给条件、各种数据结构以及算法之间的联系，这样才能更好地完成题目，并达到提高自己的目的。

这篇论文仅仅是从一类很常见的问题——约瑟夫环（也称 Josephus 排列）问题出发，并由此引申出数据结构的综合运用。对于形式多样的信息学问题来说，数据结构的综合运用只是解题策略中的一个小方面，但是如果我们对待每个问题、算法、数据结构等，都能深入发掘它与其它事物的联系，那么我们就可以自然而然地建立起知识网络，在必要的时候综合运用。而这对于我们的学习、研究将大有帮助。

特别需要强调的是：本文提到"数据结构的综合运用"，这里的综合并不单单是指形式上的，更重要的是指思想（即内涵）的综合。只要在思想上体现出两种或多种数据结构的优点，在操作时发挥出它们的优点，就已经从根本上达到了综合的目的。

从圆桌问题谈数据结构的综合运用

例．圆桌问题*(99 年安徽省赛题)*

题目：圆桌上围坐着 $2n$ 个人。其中 n 个人是好人，另外 n 个人是坏人。如果从第一个人开始数数，数到第 m 个人，则立即处死该人；然后从被处死的人之后开始数数，再将数到的第 m 个人处死……依此方法不断处死围坐在圆桌上的人。试问预先应如何安排这些好人与坏人的座位，能使得在处死 n 个人之后，圆桌上围坐的剩余的 n 个人全是好人。

输入：文件中的每一行都有两个数，依次为 n 和 m ，表示一个问题的描述信息，$n \le 32767$ ，$m \le 32767$ 。

输出：依次输出每一个问题的解。每一个问题的解可以用连续的若干行字符来表示，每行的字符数量不超过 50。但是在一个问题的解中不允许出现空白字符和空行，相邻的两个问题的解之间用空行隔开。用大写字母 G 表示好人，大写字母 B 表示坏人。

解法：

思想：模拟实际过程，寻找前 n 个被"处死"的人的位置（注：此处插入图示 1）

1． 普通解法——线性表"查找"法

1 用顺序存储结构实现

用数组记录当前所有未被处死的人在原来的位置，初始值为 1..2n。可根据前一个被处死的人在数组中的位置（即下标）直接定位，找到下一个应该被处死的人在数组中的位置，然后删去，并将它后面的元素全部前移一次。*（注：此处插入图示 2，并分析优缺点）*

如果我们将找下一个该被处死的人的操作简称为"找点"，将删除一个人后要进行的操作称为"去点"，可以看出：顺序存储结构的优点是"找点"时，可以由现在被处死的人的位置直接计算并在数组中精确定位；而缺点也很明显，就是"去点"时，都需要把它后面所有的元素整体移动一次，时间复杂度为 O(n)。所以应用顺序存储结构，程序的整体时间复杂度是 $O(n^2)$。

2 用链式存储结构实现

用链表记录当前所有未被处死的人在原来的位置，初始值为 1..2n。每处死一个人后，只要将这个结点直接从链表中删去即可，然后指针后移 (m-1) 次，找到下一个应该被处死的人。*（注：此处插入图示 3，并分析优缺点）*

链式存储结构的优点是"去点"时只要修改应该被删除结点的父结点指针指向就可以了；缺点是"找点"时，需要移动 (m-1) 次定位指针，

所以应用链式存储结构，程序的整体时间复杂度是O(nm)。

从哲学角度分析，“找点”和“去点”是存在于程序和数据结构中的一对矛盾。应用顺序存储结构时，“找点”效率高而“去点”效率低；应用链式存储结构时，“去点”效率高而“找点”效率低，这都是由数据结构本身决定的，不会随人的主观意志存在或消失。这就表明“找点”和“去点”的时间复杂度不会同时降为O(1)。我们希望有这样一种数据结构，在实现“找点”和“去点”时，使复杂度降到尽量低，在综合考虑顺序存储结构和链式存储结构的特点之后，我们设想出这样一种数据结构模型（*注：插入图示4“思想模型”*），总体思想就是在较好地实现“直接定位”的基础上，尽量避免大量元素移动。因为小规模的数据移动和指针移动，时间都可以接受，所以从总体上来说，这种数据结构的时间复杂度不会太高。实现时，我们将上面的数据结构模型做了一些小小的变动，并提出改进解法，即“优化直接定位”法。

2. 改进解法——“优化直接定位”法

设计出的存储结构如图所示：其中group表示将原来的数据分为几段存储；每一段的开头记下的amount值表示此段中现有元素的个数。随程序的运行，amount值是不断减小的。（*注：先显示图示4“实际模型”；然后手工删除，伴随讲解*）

“优化直接定位”法较好的体现出“直接定位”的思想，而且由于将所有的结点分为若干段之后，每次删除一个结点后，需要移动的结点数相对而言不是很多，这样就使程序效率大大提高，且m越大，这种效果越明显。

这种分段式数组可以看作是**链式**存储结构和**顺序**存储结构的结合产物，它兼具这两种存储结构的优点。

请注意，我们这里提到“结合产物”借用生物学中的部分思想——子代因为遗传作用而具有亲代的某些特征，同时又因为变异作用而与亲代存在差别（当然，我们希望这种变异总是向着好的方向的）。我们设计出的综合的数据结构应该继承了其“亲代”（即本来的未经变化数据结构）的优点，而摒弃它们的缺点。

运用了这种存储结构后，程序效率显著提高，可参见改进前后程序效率比较的表格。（*注：插入表格*）

引申

（*注：插入“引申”*）

- 横向延伸——约瑟夫环类的问题

  如：《翻牌游戏》、《猴子选大王》

- 纵向延伸——数据结构的综合运用

  在解决一些数据规模较大的题目时有很好的应用。如《隐藏的码字》*（IOI'99）*。在解决这道题目时，如果能建立起链式和顺序相结合的数据结构，程序效率就比较高。

  链式和顺序相结合的数据结构实现简单，效果显著，应用比较广泛。当然还有其它的结合方式，比如二叉堆和顺序结构的一一映射（单射），在解决某些问题时有非常好的效果。

小结

在计算机竞赛中成绩斐然的徐宙同学曾在他的论文《谈网络式思维方式及其应用》中写道"网络式思维方式的核心是**联系**"。在做题目时，我们也应该深挖题目所给条件、各种数据结构以及算法之间的联系，这样才能更好地完成题目，并达到提高自己的目的。

本篇论文仅仅是从一类很常见的问题——约瑟夫环（也称 Josephus 排列）问题出发，并由此引申出数据结构的综合运用。对于形式多样的信息学问题来说，数据结构的综合运用只是解题策略中的一个小方面，但是如果我们对待每个问题、算法、数据结构等，都能深入发掘它与其它事物的联系，那么我们就可以自然而然地建立起知识网络，在必要的时候综合运用。而这对于我们的学习、研究将大有帮助。

最后特别需要强调的是：本文提到"数据结构的综合运用"，这里的综合并不单单是指形式上的，更重要的是指思想（即内涵）的综合。只要在思想上体现出两种或多种数据结构的优点，在操作时发挥出它们的长处，就已经从根本上达到了综合的目的。