约定：凡是用(*和*)括住的都不是发言内容!!!

No.1

(*动画结束*)大家好！我是南京市金陵中学的刘一鸣。今天我要和大家讨论的就是：一类搜索问题的优化思想——数据的有序化。

No.2

数据有序化的思想，就是将杂乱的数据，通过简单的分类和排序，变成有序的数据，从而加快搜索的速度。

(*点击超级链接“为什么要进行数据有序化”，goto No.3*)

No.3

为什么要数据有序化呢？

我们平时遇到的搜索问题，其数据往往有一大特点，那就是杂乱无章，毫无次序可言。

(*点击鼠标，杂乱数据图出现*)就是杂乱的数据。

(*点击鼠标，有序数据图出现*)而这则是有序的数据。

但是，在同一个题目中，应用的算法相同，而数据的有序程度不同，程序的效率往往会有较大的差异。

下面，我就给大家举一个例子，“装箱问题”。

No.4

请大家仔细看题目，注意，这里和一般的“装箱问题”不同的地方就在于，题目是要求“放满”集装箱，也就是说，货物体积总和必须恰巧等于集装箱总体积。

No.5

这里提供了两种算法的时间比较，我们不难发现，第2种算法在多数情况下运行得很好，而第1种算法则不很理想。

No.6

两个程序效率不同的原因在哪里？

(*点击鼠标，图表出现*)主要的原因是，我们在使用最优性剪枝的时候，最希望的就是能较早地得到一个逼近最优解的较优解。

(*点击鼠标，"最优解"出现*)这里就是最优解。

(*点击鼠标,"不理想解"出现*)这里是不太理想的初始解,不排序而直接搜索,很可能会产生这种初始解。

(*点击鼠标，"最理想解"出现*)这里是最理想的初始解，不排序而直接搜索和先排序再搜索都可能会产生这种初始解，不过后者的概率似乎略大。

(*点击鼠标，"较理想解"出现*)这里是较为理想的初始解，先排序再搜索，很可能会产生这种初始解，这就是它的优势所在。

No.7

当然，数据有序化的优点并不仅仅是这些。首先，对于大多数数据，它都有良

好的优化效果，不过也不乏专门针对它的数据；其次，它实现起来很简单，对于刚才那个问题，在搜索前加上一个冒泡排序，只用加上几行就可以了；再其次，使用这种方法不会与其它优化方法形成冲突，甚至会为它们创造便利。所以，不难看出，数据有序化在实际应用中是大有裨益的。

No.8

数据有序化大致可以分为两种。

第1种就是“预处理阶段的数据有序化”(*点击对应的HyperLink，goto No.9*)

No.9

(*点击鼠标*)

一般来说，我们解决一个问题，都是读入数据以后直接进行数据的加工。(*点击鼠标，直至"加工"出现*)

预处理阶段的数据有序化，就是在加工之前多一个数据的处理过程，把它们由杂乱的排成有序的。(*点击鼠标，直至第2个箭头出现*)

下面，我以IOI2000的"积木搭建"为例具体讲解。

No.10

这道题目的题意大家应该比较熟悉了，我就不再多讲了。

传统的做法，是基于3维空间的算法，大家可以看看Wang Renshen同学的解题报告，这种方法虽然容易想到，但是效率并不高，有些官方测试数据甚至会超时。

现在，我们尝试在预处理阶段对数据进行有序化处理，来优化搜索。

No.11

先对构型的数据进行有序化处理。

(*点击鼠标*)将构型的所有小方块按照它们在空间中的顺序排序并编号。

(*点击2次鼠标*)用一个集合[1,v]表示构型。

(*点击2次鼠标*)这样，就将原来的3维几何体转化成了一个1维的集合。

No.12

然后，我们对积木的数据进行有序化处理。

(*点击鼠标*)枚举所有能够插入构型的积木。

(*点击2次鼠标*)用积木所包含的方块的编号组成的集合分别表示每块积木。

(*点击3次鼠标*)这样，一个积木可以放进构型里，就可以用一个集合是否包含于另一个集合来表示。

(*点击2次鼠标*)

No.13

下面，我们看看怎样从一个构型里挖去一块积木。

这是一个构型，

(*点击鼠标*)我们用[1,10]表示。现在，我们挖去一块积木，

(*点击鼠标，并将鼠标指针指向浅绿色区域*)大家请看，浅绿色区域代表挖掉的一块积木{3,6,7,9}，那么，剩下的构型就是集合{1,2,4,5,8,10}。

(*点击 2 次鼠标*)这个操作，数据有序化处理以后，我们可以用集合的减运算来表示。(*点击鼠标*)

No.14

我们现在对于剩下的那个构型

(*点击 3 次鼠标*)还想继续放一块积木

(*点击 3 次鼠标*)就是这块{4,5,7,8}。我们发现，{4,5,7,8}并不包含于{1,2,4,5,8,10}，

(*点击鼠标*)所以我们判定，

(*点击 2 次鼠标*)积木不能放入构型。

No.15

最后看看积木的冲突的判定

(*点击 2 次鼠标*)不难看出，左右两个积木单独放，都能放进构型

(*点击 2 次鼠标*)但是，这两个积木同时放，是存在冲突的，就是第 7 号方块(*鼠标指向 7 号方块*)

(*点击鼠标*)转化为集合表示，我们会发现，冲突的积木的交集不为空集。

No.16

至此，预处理阶段的数据有序化处理全部完毕，大家可以看一看有序化前后的比较。(*点击鼠标*)

我们现在已经成功地将 3 维几何体转化成了 1 维的集合，剩下来的事情，就是对集合简单地进行搜索，这个算法很基本，也很简单，只要 DFS 就可以了。

事实上，这道题目的成功解决，主要就是在预处理方面下了功夫，数据的有序化使问题的数学模型得到了精简。(*点击鼠标*)

(*点击鼠标，出现"Return"按钮，点击返回 No.8*)

No.17

接下来，我们再来看看"实时处理阶段的数据有序化"

传统的方法一般在计算出一些数据以后，(*点击鼠标*)，直接加工或保存。(*点击 4 次鼠标*)

而实时处理阶段的数据有序化，就是在计算出数据以后，先将其转化为有序的数据。(*点击 2 次鼠标*)，然后分别加工或保存，(*点击 2 次鼠标*)，在这时可能会发现一些无用的数据，就可以直接舍弃(*点击 2 次鼠标*)，这是它的额外的一种好处。

在这里，我们经常需要用到数据的最小表示法。

最小表示法是一种基于数据有序化思想的方法。其基本思想就是，对于同构的一类数据，在保存的时候只将最小的一个存储。

No.18

传统的表示方法，就是得到一个合法状态以后，先得到所有的同构状态，(*点击 2 次鼠标*)，如果这些状态中有一个已经保存，(*点击 3 次鼠标*)，则 S 可以舍弃，(*点击鼠标*)，否则保存 S(*点击 2 次鼠标*)。

No.19

而最小表示法，在得到一个合法状态以后，先将其转化为最小表示(*点击 2次鼠标*)，如果最小表示已经保存，(*点击 2 次鼠标*)，则舍弃(*点击鼠标*)；否则，将最小表示保存(*点击 2 次鼠标*)。

最小表示有一个重要性质：唯一性。每一种状态都有且只有一个最小表示。这个性质对于搜索的优化是很有帮助的。下面，我以一道例题来详细讲解。

No.20

这是一个"N 皇后问题"的改进版，请大家仔细看题目。

No.21

首先要解决的是状态的表示，这里使用的是一种大家都很熟悉的方法，就是把 2 维的棋盘转化为 1 个 n 元组表示。(*点击 3 次鼠标*)

No.22

再来看看翻转、旋转的具体过程。以 n=5 为例，首先是以铅垂线为轴的翻转，(*点击鼠标*)，左边这个棋盘是一个合法的棋盘，经过翻转，得到右边这个棋盘，(*点击鼠标*)。它们的 n 元组表示不难求出，(*点击 3 次鼠标*)，所以，我们很容易地求出了以铅垂线为轴的翻转方法。(*点击鼠标*)

No.23

依照以上的方法，我们还可以求出以水平线为轴的翻转(*点击鼠标*)

以对角线为轴的翻转(*点击鼠标*)，这里的 bi 表示第 i 行的皇后所在的列。

还有 3 种旋转过程(*点击鼠标*)。

No.24

现在，我们使用最小表示法。

一种方法就是按部就班地生成状态，转化成最小表示，再判断。(*点击 7 次鼠标*)

其实，最后保存的只有最小表示，所以，不是最小表示的解可以在发现以后就立即回溯。

在枚举的过程中(*点击鼠标*)，如果发现由当前状态不可能生成最小表示，则回溯(*点击 3 次鼠标*)，否则继续枚举(*点击鼠标*)。这就提供了一个新的剪枝(*点击鼠标*)。

No.25

由翻转、旋转的具体过程可知，当前搜索到的状态如果满足最小表示，必须符合这些条件(*点击 2 次鼠标*)。这是一个比较强的剪枝条件，在实际应用中能剪掉一大半的无用枝杈。

No.26

再看看这幅图，我们发现，利用新的剪枝条件，能够找到的解一定是最小表示的解。(*点击 2 次鼠标*)

由最小表示的唯一性，能够搜索到的、并满足最小表示的解一定是不同构的。

所以，判断同构的过程可以省略，已经搜索到的状态也因此可以不保存。(*点击鼠标*)空间复杂度因此大幅下降。

No.27

这是数据有序化前后的时空复杂度对比，S 是合法解的集合，(*用鼠标指向"|S|"*)。最小表示的优势是显而易见的。

No.28

下面我们来比较一下两种实现方法。(*点击鼠标*)

预处理阶段的数据有序化，时间上的耗费小一些，但对空间的要求较高。

而实时处理阶段的数据有序化，更灵活一些，数据可以即时处理，所以空间要求小一些。但是，如果搜索树的结点较多，它不一定会很理想，因为它可能会重复处理某些相同的结点。

但是，这两种方法实际上是优势互补的。所以，在应用的时候，我们可以将两种方法并用，扬长避短，达到更好的效果。

No.29

最后，我们对今天的研究做一个总结。

我们今天所讨论的，就是将混乱无序的数据，通过简单的方法，转化成为符合科学美的有序数据。科学本身就是一种美，我们努力创造出的符合科学美的数据，有时会令我们事半功倍。

也许大家会说，今天所讨论的题目，我所讲的方法都不是最好的方法，许多方法能使程序运行得更快。但是，我在这里要说的是一个性价比的问题，今天我所讲的优化方法，都只要在原来的程序作一些修改就可以了，不必将原来的程序推翻从头写，很经济，很划算。我们都知道，不论是在竞赛中，还是在生活中，解决一个问题，重要的是尽快设计出解决方案，进而获得答案，而不是让我们的程序运行的时间最短，只要能在限定时间内解决问题，就是成功。所以，只有合理的选择，才能获得最大的性价比。

No.30

我的发言到此结束，谢谢大家！