深度优先搜索问题的优化技巧

重庆一中 黄晓愉

深度优先搜索的优化技巧

在深度优先搜索中如何运用题目中的约束条件为我们提供剪枝是影响程序效率的关键。而搜索的顺序和搜索的对象对于这一点是十分重要的。

搜索顺序的选择

我们先来看一道比较简单的题目：

(zju1937)

已知一个数列 a0,a1.....am 其中

a0 = 1

am = n

a0 < a1 < a2 < ... < am-1 < am

对于每个 k(1<=k<=m),ak=ai+aj (0 <= i, j <= k-1)，这里 i 与 j 可以相等。现给定 n 的值，要求 m 的最小值

简单的分析

- 依次搜索是很容易想到的方法，而对于每个数的取值，我们显然可以采用从小到大搜索和从大到小搜索两种搜索方法。

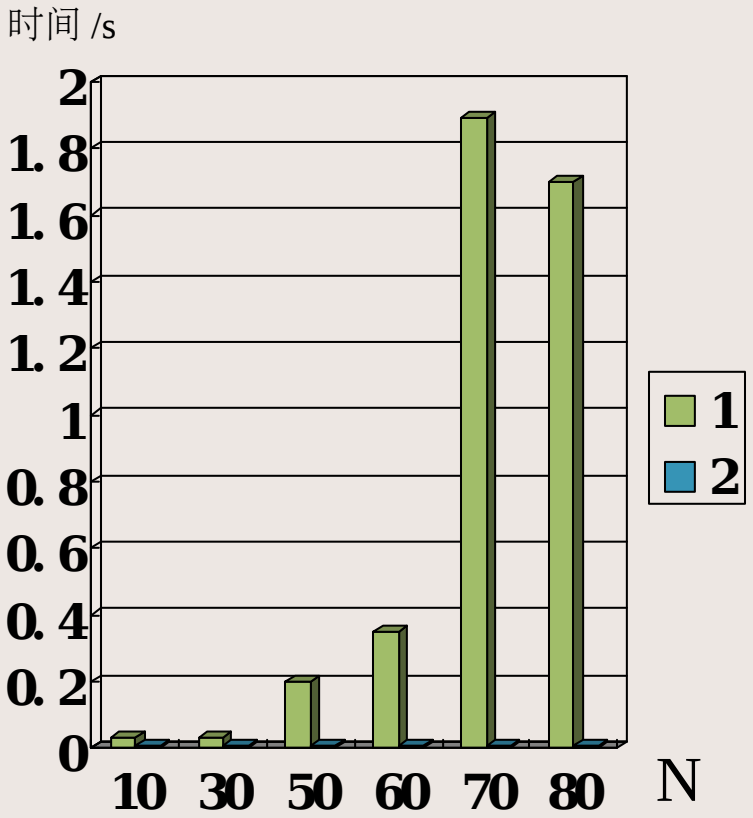
由于题目要求的是 m 的最小值，也就是需要我们尽快构造出 n，所以每次构造的数应当是尽可能大的数 。

不同搜索顺序效率比较

两种搜索顺序比较:

1、从小到大搜索每个数值

2、从大到小搜索每个数值



显然，不同的顺序导致了程序效率的不同

以往比赛中的情况

- IOI2000 中的 BLOCK

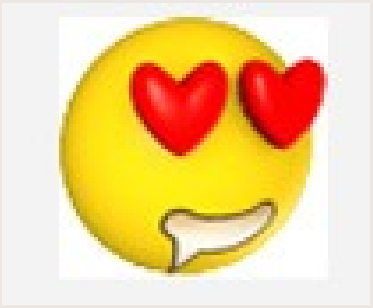
将木块从大到小经过旋转和反转后，依次放入进行搜索

满分！！

- NOI2005 中的智慧珠

将珠子从大到小进行搜索，不加任何其他剪枝

90 分！！



搜索对象的选择

- (USACO － weight)
- 已知原数列 a1， a2……an 中前 1 项，前 2 项，前 3 项……前 n 项的和，以及后 1 项，后 2 项，后 3 项……后 n 项的和，但是所有的数据都已经被打乱了顺序，还知道数列中的数存在集合 S 中，求原数列。当存在多组可能数列的时候求左边的数最小的数列。
- 其中 n<=1000,S∈{1..500}

一个例子

假如原数列为 1 1 5 2 5， S={1， 2， 4， 5}
那么知道的值就是：

(1 2 7 9 14 5 7 12 13 14)

- 1 = 1
- 2 = 1+1
- 7 = 1+1+5
- 9 = 1+1+5+2
- 14 = 1+1+5+2+5
- 5 = 5
- 7 = 2+5
- 12 = 5+2+5
- 13 = 1+5+2+5
- 14 = 1+1+5+2+5

一般方法

- 从左往右依次搜索原数列每个数可能的值，然后与所知道的值进行比较。

- 但是这个算法最坏的情况下扩展的节点为 500^{1000}，这个算法

太慢了

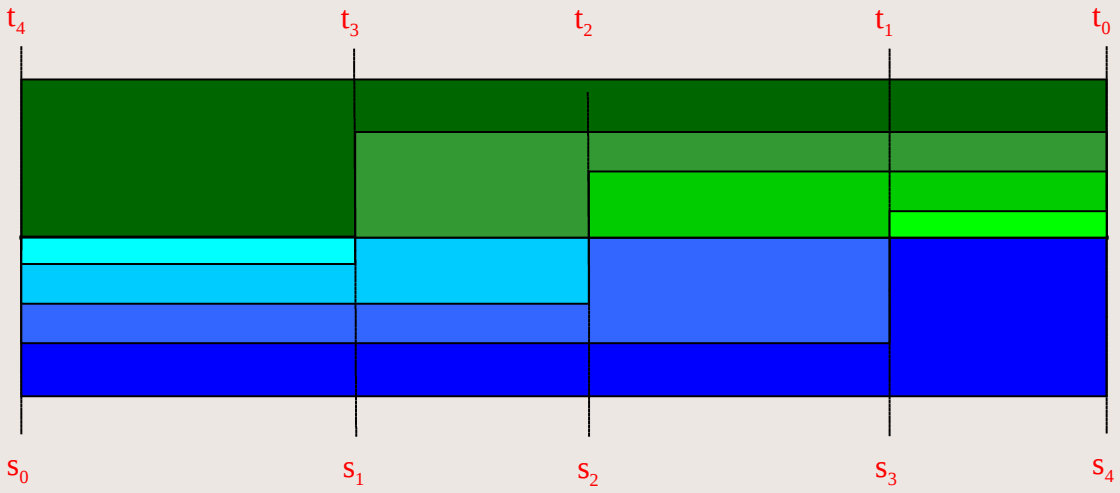
如何改进?

从已知入手分析

我们用

S_i 表示前 I 个数的和

T_i 表示后 I 个数的和



任意 I 满足： $S_i+T_{n-I}=S_n=T_n$

对题目中数据分类

集合 **A**

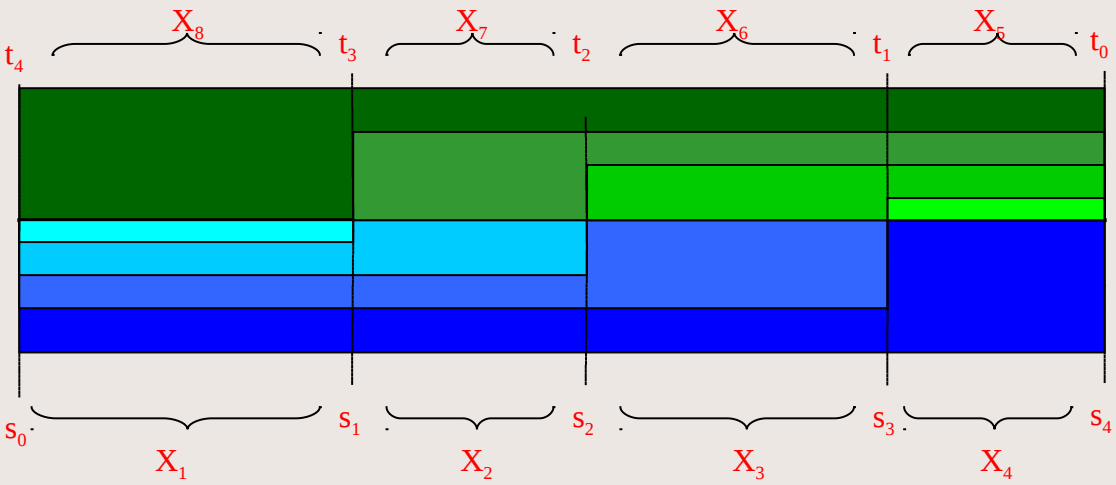
集合 **B**

分析

在集合 A 和集合 B 中：

$\{S_0<S_1<S_2<\ldots\ldots<S_n\}$

$\{T_0<T_1<T_2<\ldots\ldots<T_n\}$



S_i-S_{i-1}

T_i-T_{i-1}

$\Rightarrow Xi \in S$

在搜索中从小到大搜索每个 Si 和 Ti 的位置。

由具体分析

对于原数列：1 1 5 2 5，S=｛1,2,4,5｝

由它得到的值为：

1 2 7 9 14 5 7 12 13 14

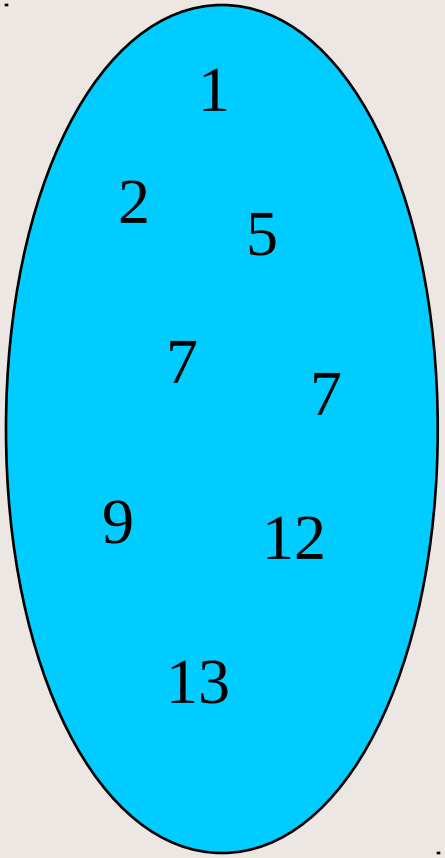
排序后为：

1 2 5 7 7 9 12 13 14 14

原数列：

1 *1* 5 2 5

这样，对于这个数据，我们已经构造出了原数列。

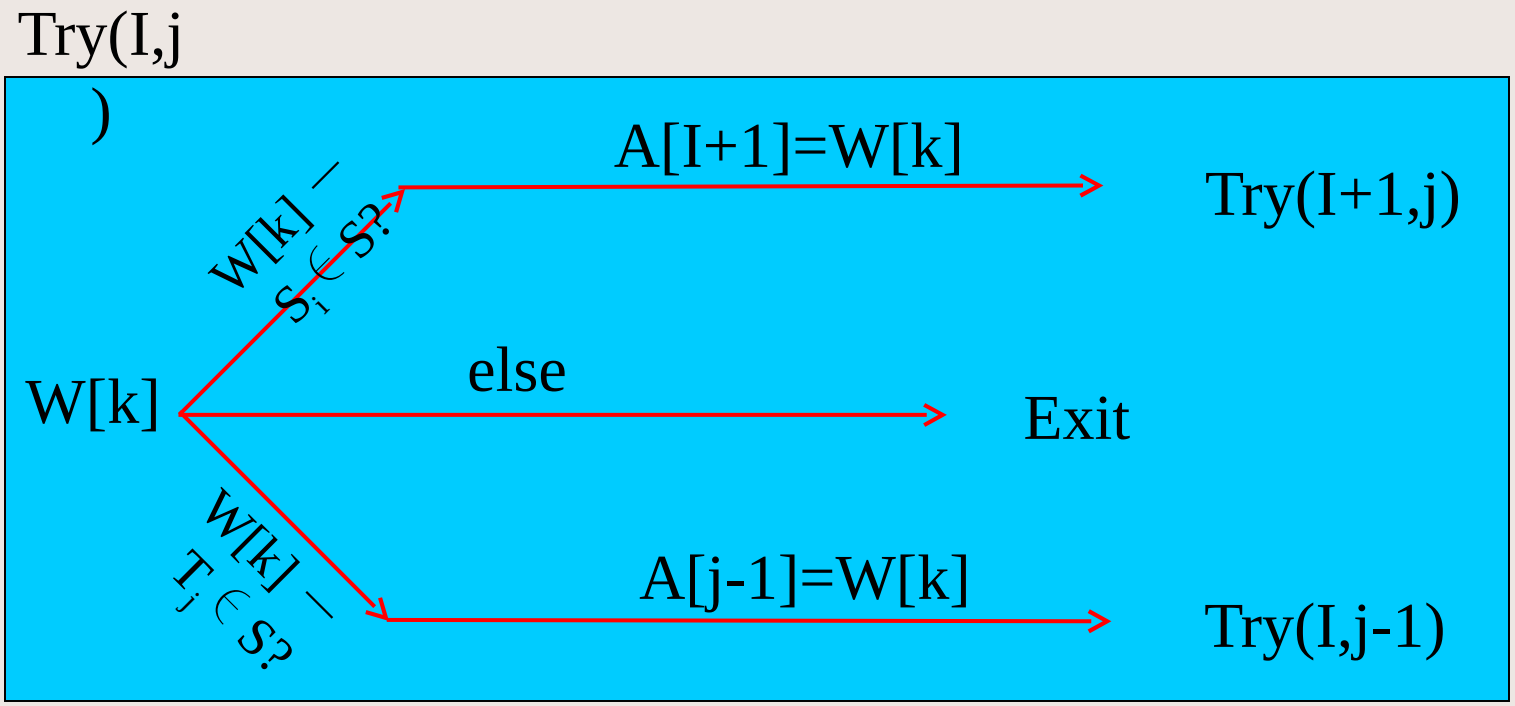


改变搜索对象

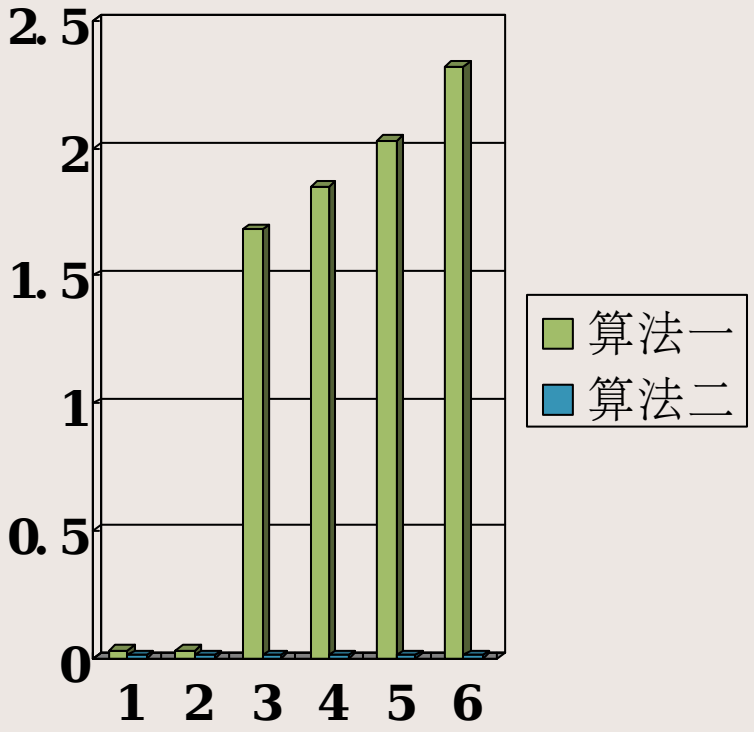
- 题目的约束条件集中在 S_i 和 T_i 中，我们改变搜索的对象，不再搜索原数列中每个数的值，而是搜索给出的数中出现在 S_i 或者 T_i 中的位置。又由于 S_{i+1} 与 S_i 的约束关系，提示我们在搜索中按照 S_i 中 i 递增或者递减的顺序进行搜索。

推而广之

当我们已经搜索出原数列的 a_1 ， a_2……a_i 和 a_n,a_{n-1}…… aj ，此时搜索排序后第 k 小的数 W[k], 只可能有两种存在的可能:



- 这个算法在最坏情况下扩展的节点为 2^{1000}(实际中远远小于这个数)，在搜索的同时可以利用 Si+Tn-I=Sn=Tn 这个约束条件进行剪枝。程序效率得到显著的提高。两个程序效率对比:



小结

- 原始的搜索方法搜索量巨大，我们通过分析，选择适当的搜索对象，在搜索量减少的同时充分利用了题目的约束条件，成为了程序的一个有利的剪枝，使题目得到较好的解决。

总结

我们在搜索得过程中，灵活得改变搜索的顺序和搜索的对象可以使程序效率得到很大的提升。

而这需要我们在做题的过程中多思考、多分析、多总结。