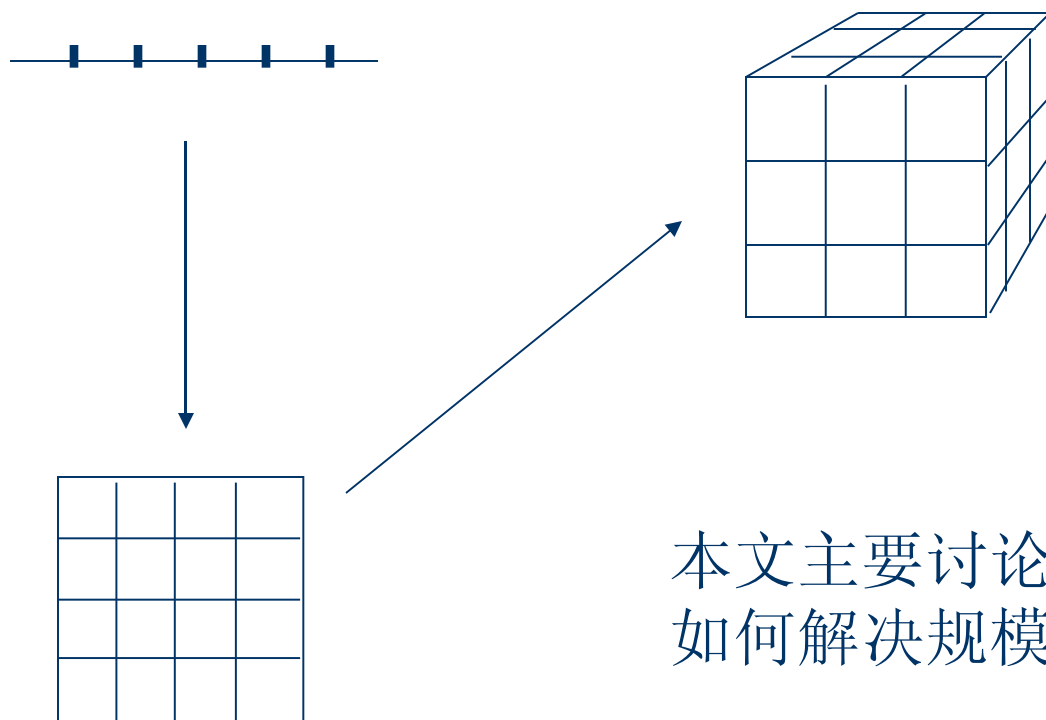


用改进算法的思想解决规模维数增大的问题

广东韶关一中 张伟达

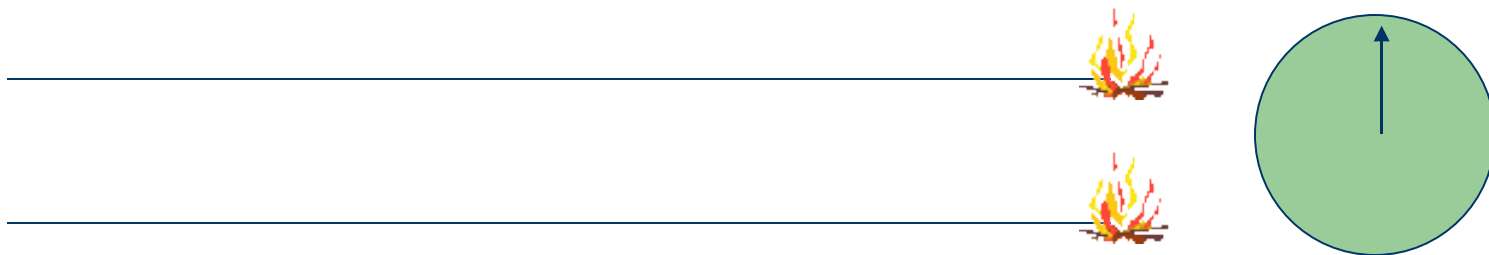
一、概述



本文主要讨论
如何解决规模维数增大的问题

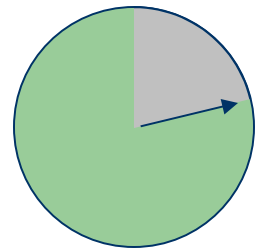
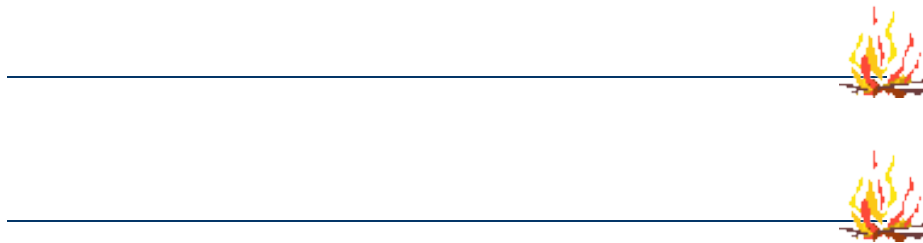
二、引子：从一道 IQ 题说起

有两根完全相同但分布不均匀的香，每根香烧完的时间是一个小时，你能用什么方法来确定一段 45 分钟的时间？



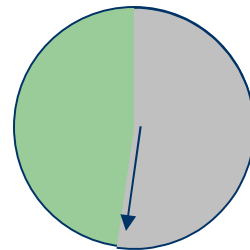
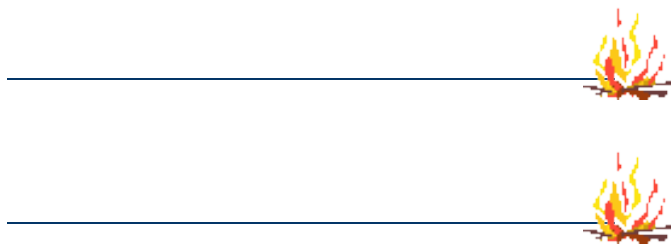
二、引子：从一道 IQ 题说起

有两根完全相同但分布不均匀的香，每根香烧完的时间是一个小时，你能用什么方法来确定一段 45 分钟的时间？



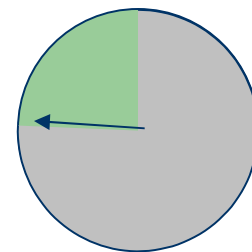
二、引子：从一道 IQ 题说起

有两根完全相同但分布不均匀的香，每根香烧完的时间是一个小时，你能用什么方法来确定一段 45 分钟的时间？



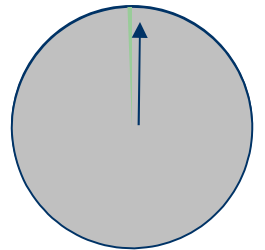
二、引子：从一道 IQ 题说起

有两根完全相同但分布不均匀的香，每根香烧完的时间是一个小时，你能用什么方法来确定一段 45 分钟的时间？



二、引子：从一道 IQ 题说起

有两根完全相同但分布不均匀的香，每根香烧完的时间是一个小时，你能用什么方法来确定一段 45 分钟的时间？




二、引子：从一道 IQ 题说起

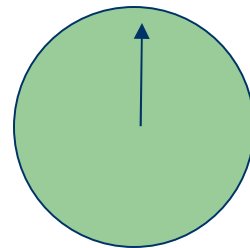
 模型 A. 不减小每根香的计时，两根香加在一起计时是 45 分钟；

显然不成立！


二、引子：从一道 IQ 题说起

-  模型 A. 不减小每根香的计时，两根香加在一起计时是 45 分钟；
- 模型 B. 只用一根香，使其计时减小，直接计时 45 分钟；
- 模型 C. 把两根香的计时都减小，使两根香加起来是 45 分钟。

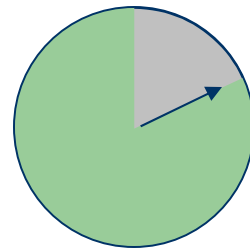
既然都必须把计时减小，不难看出有两头烧的方法：




二、引子：从一道 IQ 题说起

-  模型 A. 不减小每根香的计时，两根香加在一起计时是 45 分钟；
- 模型 B. 只用一根香，使其计时减小，直接计时 45 分钟；
- 模型 C. 把两根香的计时都减小，使两根香加起来是 45 分钟。

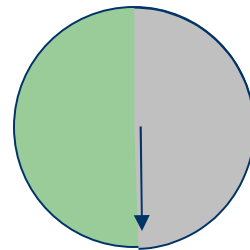
既然都必须把计时减小，不难看出有两头烧的方法：



二、引子：从一道 IQ 题说起

-  模型 A. 不减小每根香的计时，两根香加在一起计时是 45 分钟；
- 模型 B. 只用一根香，使其计时减小，直接计时 45 分钟；
- 模型 C. 把两根香的计时都减小，使两根香加起来是 45 分钟。

既然都必须把计时减小，不难看出有两头烧的方法：



二、引子：从一道 IQ 题说起

 模型 A. 不减小每根香的计时，两根香加在一起计时是 45 分钟；

 模型 B. 只用一根香，使其计时减小，直接计时 45 分钟；

模型 C. 把两根香的计时都减小，使两根香加起来是 45 分钟。
。

如果我们两头一起烧，用一根香就能够计时 30 分

钟
再看模型 B，显然 $30+30$ 是不可能等于 45 分钟
的

排除！

二、引子：从一道 IQ 题说起

C. 模型

+ 两头烧的方法

$(\&)\@ \# (\@^\wedge$

改进两头烧的方法

:

能够计时 1 小时  计时 30 分钟

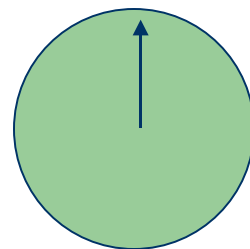


能够计时间 t  计时 $t/2$



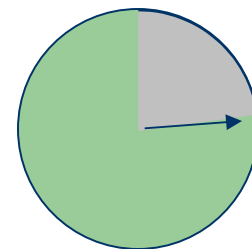
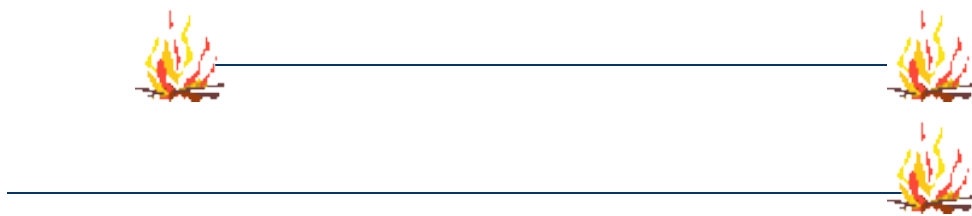
二、引子：从一道 IQ 题说起

1。分别点燃第一根的两头和第二根的一头；



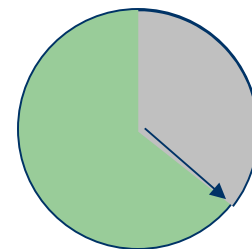
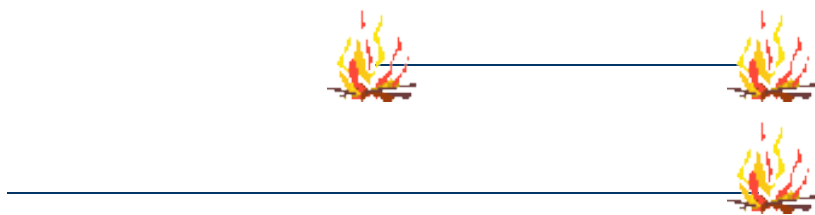
二、引子：从一道 IQ 题说起

1。分别点燃第一根的两头和第二根的一头；



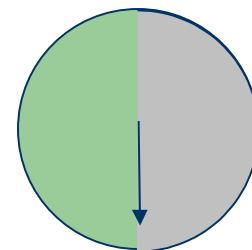
二、引子：从一道 IQ 题说起

1。分别点燃第一根的两头和第二根的一头；



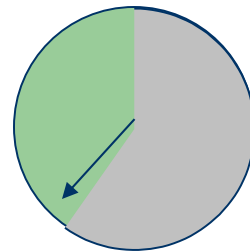
二、引子：从一道 IQ 题说起

- 1。分别点燃第一根的两头和第二根的一头；
- 2。第一根烧完的时候，已经过了 30 分钟；第二根还剩 30 分钟，点燃第二根的另一头；



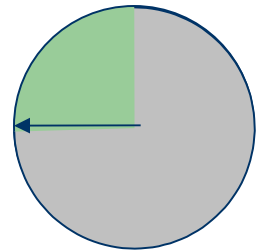
二、引子：从一道 IQ 题说起

- 1。分别点燃第一根的两头和第二根的一头；
- 2。第一根烧完的时候，已经过了 30 分钟；第二根还剩 30 分钟，点燃第二根的另一头；



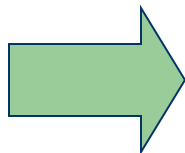
二、引子：从一道 IQ 题说起

- 1。分别点燃第一根的两头和第二根的一头；
- 2。第一根烧完的时候，已经过了 30 分钟；第二根还剩 30 分钟，点燃第二根的另一头；
- 3。当第二根也烧完了，即时间又过了 15 分钟。那么我们计出的总的时间就为 45 分钟了。



二、引子：从一道 IQ 题说起

C. 模型
+ 两头烧的方法
—————
(*&)@#(@^



C. 模型
+ 改进的两头烧的方法
—————
成功解决问题

二、引子：从一道 IQ 题说起

小结一：做这类问题的主要流程

1. 找出原始解法和可能改进的方向（即分析成 A、B、C 模型）
2. 分析算法的原理（由烧一根香计时半小时，引申为烧剩 t 的时候点两头就能计时 $t/2$ ）；
3. 改进算法（改进的过程中，往往不是依靠算法改进算法本身，反而是利用算法的内涵、实质，结合问题，构造算法）；
4. 解决问题（我们得出了正确的解法）。

三、改进算法的途径

1. 直接增加算法的规模，解决问题
2. 用枚举处理增加的规模，从而解决问题
3. 用贪心解决增加的规模，从而解决问题
4. 多种途径的综合运用

为了能够简单明了地解释改进算法的途径，我们直接进入第 4 种情况，用一道例题详细讲解可以如何改进算法解决问题。

【例题】 Team Selection (Balkan OI 2004 Day1)

【题目大意】 IOI 要来了， BP 队要选择最好的选手去参加。幸运地，教练可以从 N 个非常棒的选手中选择队员，这些选手被标上 1 到 N ($3 \leq N \leq 500000$)。为了选出的选手是最好的，教练组织了三次竞赛并给出每次竞赛排名。每个选手都参加了每次竞赛并且每次竞赛都没有并列的。当 A 在所有竞赛中名次都比 B 前，我们就说 A 是比 B better。如果没有人比 A better，我们就说 A 是 excellent。求： excellent 选手的个数。

【例题】 Team Selection (Balkan OI 2004 Day1)

如数据：

10

2	5	3	8	10	7	1	6	9	4
1	2	3	4	5	6	7	8	9	10
3	8	7	10	5	4	1	2	6	9

例如 5，没有选手次次竞赛都比 5 强，
因为 5 在第一次竞赛中只输给了 2，但是 2 又在第三次竞赛中输给了 5，所以 5 是 excellent

【例题】 Team Selection (Balkan OI 2004 Day1)

【原始思路

【原始算法】枚举每一个选手 X ，判断 X 是否 excellent。这可以通过另一重循环，枚举另一选手 Y ，判断 Y 是否比 X better。判断是容易的，我们只需要简单地判断 X 和 Y 的三次排名。

```
for(X 从 1 到 N)
    for(Y 从 1 到 N)
        判断 Y 是否比 X better
```

Time: $O(N^2)$

【例题】 Team Selection (Balkan OI 2004 Day1)

【原始思路

】『改进一』 如果让 X 依照第一次竞赛的名次循环，枚举 Y 时只需要枚举在第一次竞赛中排在 X 前面选手即可。

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 从第一次竞赛的第 1 名到第一次竞赛的第 (X-1) 名  
    )  
        判断 Y 是否比 X better
```

Time: $O(N^2)$

【例题】 Team Selection (Balkan OI 2004 Day1)

【原始思路

】 〔改进二〕 如果 Y 不是 excellent （因为有 Z 比 Y better ）， 当我们检查 X 是否 excellent 时， 我们只需要检查了 Z 是否比 X better ， 可以不检查 Y 。

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

Time: $O(NK)$ （设 K 是 excellent 选手的个数）

【例题】 Team Selection (Balkan OI 2004 Day1)

【原始思路

〔原始思路小结〕这里的原始算法是直接根据原始模型模拟出来的，改进一和改进二都单纯地根据原始算法的设计缺陷来“改进”（这个改进没有利用问题的本质内容，不是本文所要阐述的“改进”），所以最后的时间复杂度没有质的进展。

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

子问题： N 个选手进行两次竞赛， **better** 和 **excellent** 的定义和原题一样，问有多少 **excellent** 选手？

为了方便说明，我们设第一次竞赛排名依次为 A_i （表示第一次竞赛的第 i 名是 A_i ）， A_i 号选手在第二次竞赛中的排名的为 $B[A_i]$ （注意 $B[A_i]$ 与 A_i 的含义不同）。

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： A_1 A_2 A_3 A_4 A_5 A_6 A_7

Excellent

:

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： A_1 A_2 A_3 A_4 A_5 A_6 A_7

Excellent A_1

:

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： A_1 A_2 A_3 A_4 A_5 A_6 A_7

Excellent

A_1

:

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： A_1 A_2 A_3 A_4 A_5 A_6 A_7

Excellent

A_1 A_2

:

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： A_1 A_2 A_3 A_4 A_5 A_6 A_7

Excellent

A_1 A_2

:

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： A_1 A_2 A_3 A_4 A_5 A_6 A_7

Excellent

A_1 A_2 A_3

:

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： A_1 A_2 A_3 A_4 A_5 A_6 A_7

Excellent

A_1 A_2 A_3


:

【例题】 Team Selection (Balkan OI 2004 Day1)


【降维思路

】 这样的子问题做法仍然可以参照改进三：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名 )  
    for(Y 枚举当前已知的 excellent)  
        判断 Y 是否比 X better
```

第 1 次竞赛： $A_1 A_2 A_3 A_4 A_5 A_6 A_7 \dots A_i$ 

Excellent $A_1 A_2 A_3 \dots A_{i-1}$



：

A_i 只要比 $A_1 A_2 \dots A_{i-1}$ 任何一个大就不是
Excellent

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

A_i 只要比 $A_1 A_2 \dots A_{i-1}$ 任何一个大就不是

Excellent



比较 $\min(A_j)$ 与
 A_i

〔改进三〕 对于两次竞赛的情况，当 $X=A_i$ 时，设 $best_i = \min(B[A_j])(j < i)$ ，则 $B[A_i]$ 只需要与 $best_i$ 比较即可。

【例题】 Team Selection (Balkan OI 2004 Day1)

【降维思路

〔降维思路小结〕 在这次分析中，我们从两次竞赛——简化后的问题出发，通过简单的观察和思考，得出了改进的算法，但是优化后的算法要应用到三维的情况还有很长的路要走。

【例题】 Team Selection (Balkan OI 2004 Day1)

【扩展思路

改进三本质：

X 依照第一次竞赛的名次循环，

$X=A_i$ 时，因为 $\text{best}_i = \min(B[A_j])(j < i)$ 中， $j < i$ ，这样就保证了 A_j 在第一次竞赛中名次一定比 A_i 前

如果 $\text{best}_i < B[A_i]$ ，这样就保证了 A_j 在第二次竞赛中名次比 X 前；

总之则必然有 A_j 比 A_i better。

【例题】 Team Selection (Balkan OI 2004 Day1)

【扩展思路

】〔改进四〕

1. $j < i$

2. $B[A_j] < B[A_i]$

3. 比较 $\min(C[A_j])$ 和 $C[A_i]$ （ C 表示第三次竞赛排名）

A_i 是
excellent

等价于

不存在这样的 j
 $j \in \{j \mid j < i, B[A_j] < B[A_i]\}$
 $\min(C[A_j]) < C[A_i]$

【例题】 Team Selection (Balkan OI 2004 Day1)

【扩展思路

【改进五】 通过改进四，我们观察到，我们要在一个数列中找 B 小于某个数的元素，又要找出所有这些元素的 C 的最小值。

也就是说，我们需要做这样的操作：

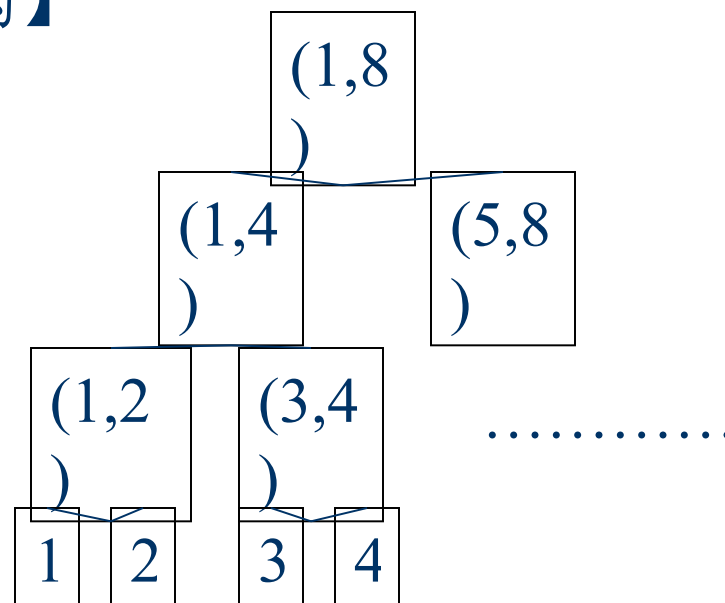
插入： 加入一对数 (**key,value**) 到数据结构中。

查询： 查询 **key** 小于 X 的最小的 **value**
显然，这里的 **key** 代表 $B[A_i]$ ， **value** 代表 $C[A_i]$ 。

【例题】 Team Selection (Balkan OI 2004 Day1)

【改进五的实现：优化数据结构】

我们构建如图的数据结构，也就是二叉检索树，每个节点表示某一个 key 的区间，节点的值是该区间内 value 的最小值。



在这样的数据结构的帮助下，插入和查询操作都能在 $O(\log N)$ 的时间复杂度下完成

【例题】 Team Selection (Balkan OI 2004 Day1)

〔最终算法〕这样我们就能很清晰地得出优化的解法：以第一次竞赛的名次从高到低枚举 X ，以第二次竞赛名次为 key ，第三次竞赛名次为 $value$ 。对于每个 X ，只要查询区间 $[1, key]$ 的最小值 $min-value$ ，若 $min-value < value$ ，则有选手比 X better，即 X 不是 excellent。反之，若 $min-value > value$ ，说明 X 是 excellent，并把 $(key, value)$ 加入检索树。

Time: $O(N \log N)$

【例题】 Team Selection (Balkan OI 2004 Day1)

【小结】 Team Selection 一题不仅很好地反映了改进算法对解决规模维数增大的问题所起的重要作用，而且还突出了数据结构的应用对解决问题的帮助。正是印证了“算法 + 数据结构 = 程序”

四、总结

改进算法的思想究竟是什么呢？

开阔进取的思想：永不满足于现状，不断改进

创新的思想：摒弃旧思维，创造新思维

希望大家不仅在程序设计中不断进取和创新，在人生中也要有不断进取和创新

。

谢谢！