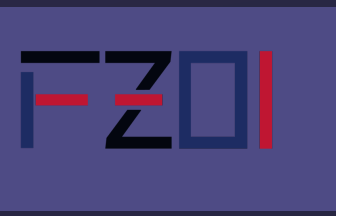
二分与分治



HenryHuang

二分查找

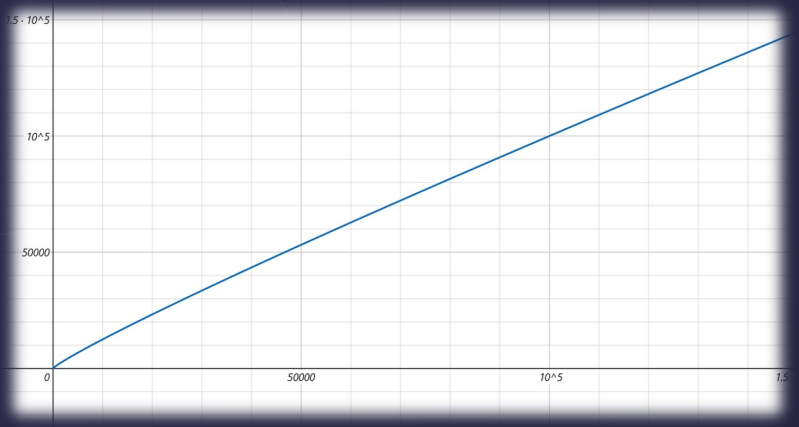
- 为什么是二分？
- 为什么不能三分呢？
- 为什么不能四分呢？
- ……

二分查找

- 事实上是可以的。
- 二分查找时，我们每次会取一个数据点进行判定，每次判定的时间为 $O(1)$，数据规模变为为原来的二分之一。
- 三分查找时，我们每次会取两个数据点进行判定，每次判定的时间为 $O(1)$，数据规模变为为原来的三分之一。
- 也就是说三分查找的时间复杂度为 $O(2\log_3 n)$。
- 类推，k 分查找的时间复杂度为 $O((k-1)\log_k n)$。

二分查找

- 我们可以通过工具将这个函数的图像画出来。
- （事实上你可以求导）
- 这里以 $O\big((k-1)\log_k 100000\big)$ 为例。
- 可以发现当 k 小时时间复杂度更小。
- 所以我们选用二分查找。



二分答案

- 本质上是将最值问题转化为了判定性问题，且最值具有单调性。

「CF371C」 Hamburgers

- 做一个汉堡需要 b,s,c 三种材料，每种材料的所需数量已经给出。
- 初始状态你拥有的三种材料数量分别为 nb,ns,nc 个，且它们的单价分别为 pb,ps,pc 元钱。
- 现在你有 r 元钱，问能制作汉堡的最大数量。

「CF371C」 Hamburgers

- 数据范围很大，不能枚举。
- 显然最大数量递增。
- 直接二分答案即可。

二分答案

- 有一类问题，是让你在最大化……的前提下，使得……最小。
- 或者是在最小化……的前提下，使得……最大。
- 或者是让最小的尽可能大，最大的尽可能小。
- 像这样的问题一般都可以使用二分答案去掉一层最值限制，从而使问题变得简单。

「NOIP2018」 赛道修建

- 给定一棵 n 个点的树，第 i 条边长度为 l_i。
- 你需要从中选出 m 条互不相交的链，使最短的一条链长度最长，并求出该值。

「NOIP2018」 赛道修建

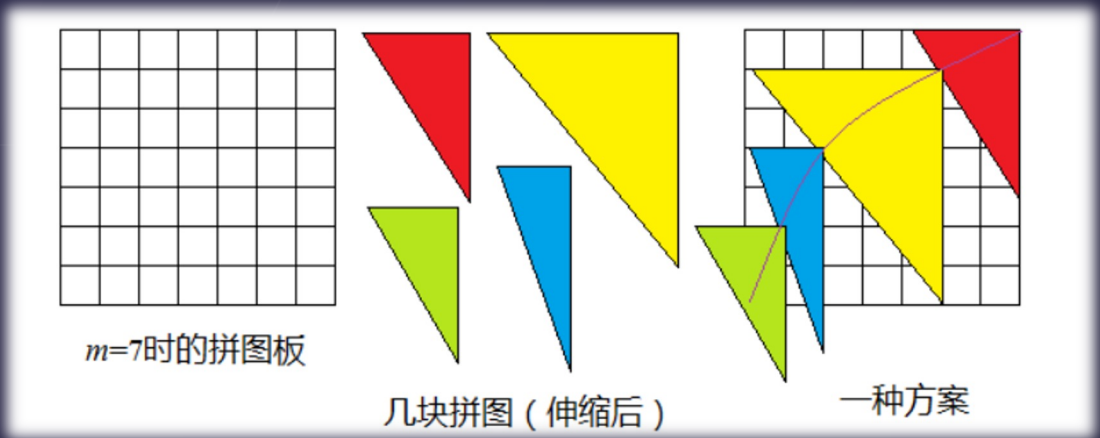
- 首先二分答案 ans，问题转化为判断能否从中选出 m 条互不相交的链，使最短的一条链长度 $\geq ans$，并求出该值。
- 不妨先考虑菊花图的情况。
  - 对于长度 $\geq ans$ 的边，我们当然乐意单独修建一条赛道。
  - 否则，为了避免浪费，我们希望找到另一条边，使得两边之和尽可能大于 ans 且贴近 ans。
- 形式化地讲，假设当前所选边长度为 a，我们想要找到 $\geq ans - a$ 且最短的另一条边与其匹配，这样是最优秀的。
- 寻找需要用到前面讲到的二分查找。

「NOIP2018」 赛道修建

- 那么考虑完整的题目，由于链互不相交，所以一个子树只能向上传递一条未完成的链。
- 注意到我们刚才做的就是儿子之间的合并，也就是两层之间的关系。
- 在做完刚才的操作后将剩余未完成的链最大的向上传递即可。
- 总时间复杂度为 $O(n\log_2^2 n)$。

「LOJ500」 ZQC 的拼图

- 给定一个 $m \times m$ 的网格，有 n 个已知大小的直角三角形，直角顶点必须在右上角且不能反转。
- 你现在你需要将这些直角三角形放在网格上，需要保证直角顶点在格点上，存在一条路径可以从网格的左下角抵达右上角，且路径上均有直角三角形覆盖。
- 现在你可以将某些直角三角形的边长放大 k 倍，求符合条件的最小正整数 k。



「LOJ500」 ZQC 的拼图

- 显然将全部三角形都放大最合算。
- 那么我们同样考虑二分答案，问题转化为将所有三角形放大 k 倍后是否满足条件。
- 那么如何判定呢？

「LOJ500」 ZQC 的拼图

- 显然一定存在一组最优解直角顶点的位置坐标在两个方向上都是单调的。
- 所以，直角三角形的摆放顺序并没有影响，因为这只会影响相对位置的改变。
- 根据这一点我们可以考虑 DP。
- 设 $f_{i,j}$ 为当前用了前 i 个三角形，横坐标为 j 所能达到的纵坐标的最大值。
- 转移式非常好推。
- 最后判断其是否大于 m 即可。

二分答案

- 到这里其实你会发现无论是二分答案或是二分查找，更多的并不表现为一种特定的算法，而是一种工具。
- 往往当你无法解决某些最值问题时，首先考虑其单调性，然后将其转化为判定性问题，往往会好做很多。
- 另提供一种万能二分答案模板：

```
while(l<=r){
        int mid=(l+r)>>1;
        if(check(mid)) ans=max(mid,ans),l=mid+1;
        else r=mid-1;
}
```

分治

- 分治的本质思想：
  - 处理子问题
  - 处理子问题之间的贡献
- 例如归并排序：
  - 将子区间内排序
  - 将二者归并
- 所以分治的关键在于我们容易处理子问题之间的贡献，或者子问题之间没有关系。

「CF1385D」 a-Good String

定义：字符串 `s` 为一个 `c-好串` （c 为一个字符）时，必须满足：

1. 当$|s|=1$，$s=c$
2. 当$|s|>1$, s 的左半部分为全为 c，右半部分为一个 `(c+1)-好串` 或者 s 的右半部分为全为 c，左半部分为一个 `(c+1)-好串`

其中 $|s|$ 代表 字符串 s 的长度。

举个例子：当 $s=$ “$cdbbaaaa$”时，s 是一个 `a-好串`

现在，给你一个字符串 s ($|s|=2^k$),问最少替换多少个字符，使其为一个 `a-好串` 。

「CF1385D」 a-Good String

- 你发现按照题意模拟，左半和右半没有任何关系。
- 所以直接分治分类讨论即可。

「CF1442D」 Sum

- 给定 n 个栈，其中元素从栈顶单调不降。
- 你可以从栈中弹出 k 个元素，求 k 个元素之和的最大值。
- $n, k \leq 3000$。
- 栈中总元素个数 $\leq 10^6$。

「CF1442D」 Sum

- 很容易想到 01 背包。
- 暴力做是 $O(n^3)$ 的。
- 但是我们并没有很好利用这里单调不降的性质。

「CF1442D」 Sum

- 考虑两个栈 a, b。
- 由于其是不降的，我们会不会每个栈雨露均沾呢？
- 答案显然是否定的。
- 因为对于一个栈，越往后取显然其能获得的价值越大。
- 也就是说，如果我们笃定要选取一个栈，最优的方式是将这个栈中的元素全部选取，除非没有足够的次数。
- 那么最后的状态一定是：若干个完全选取的栈，以及至多一个未完全选取的栈。

「CF1442D」 Sum

- 于是我们现在需要做的是除开某个栈过后，其他栈做 01 背包。
- 这个东西可以通过分治解决。

「CF1442D」 Sum

- 具体来说，这非常类似于一个二分查找的过程。
- 先将序列平均分为左边和右边两部分。
- 首先我们假设这个没选完的栈在右边，然后加入左边的所有元素，然后往右边递归。
- 然后我们也可以假设这个没选完的栈在左边，然后加入右边的所有元素，然后往左边递归。
- 回退只需要保存数组，空间与时间复杂度均为 $O(n)$。
- 这样递归到序列长度为一时，恰好是这个栈的元素未加入背包的状态，单独做背包即可。
- 总时间复杂度为 $O(n^2\log_2 n)$。

「CF1609F」 Interesting Sections

- 给定一个长度为 n 个非负整数序列 A，求有多少个区间 $[l,r]$ 满足 $\text{popcount}(\max_{i=l}^{r} a_i) = \text{popcount}(\min_{i=l}^{r} a_i)$。
- $\text{popcount}(x)$ 即 x 在二进制下 1 的个数。

「CF1609F」 Interesting Sections

- 一般这种数区间的题目都有两种方式去思考：
  - 固定一个端点计算另一个端点的数量
  - 分治，即将左右端点限制范围递归计算
- 这里我们不妨使用分治。

「CF1609F」 Interesting Sections

- 我们将区间 $[l,r]$ 递归分为 $[l,mid]$ 和 $[mid+1,r]$两个部分，考虑左端点在 $[l,mid]$ 而右端点在 $[mid+1,r]$ 时的答案。
- 注意到对于一个逐渐扩大的区间，$\max,\min$ 都是单调的。
- 这启发我们可以从大到小枚举左端点，那么此时的 $\max,\min$ 的情况为
  - 均在 $[l,mid]$ 中
  - 有一个在 $[l,mid]$ 中
  - 均不在 $[l,mid]$ 中
- 我们可以在右端点维护两个指针 p_1,p_2，表示三种情况的分界。
- 由于区间逐渐扩大，易知 p_1,p_2 单调递增。

「CF1609F」 Interesting Sections

- 第一种和第三种情况我们非常好计算贡献。
- 对于第二种情况，我们只需要在指针移动的过程中开一个桶统计即可。
- 于是这个题做完了。
- 时间复杂度为 $O(n(\log_2 n + \log_2 \max a_i))$。

CDQ 分治

- CDQ 分治是一种思想。
- 所以这里我们只讲最基础的题目。

「洛谷 P3810」陌上花开

- 给定 n 个三元组 (a,b,c)，求满足 $a_i \le a_j, b_i \le b_j, c_i \le c_j$ 的 (i,j) 对数。

「洛谷 P3810」陌上花开

- 首先我们可以将元素按照 a_i 从小到大排序，这样我们就只需要满足 b 和 c 的限制。
- 那么我们考虑分治。
- 同样，对于 $[l,r]$ 区间，递归分为 $[l,mid],[mid+1,r]$ 两个区间进行计算。
- 我们只考虑点对分别在两个区间内的答案如何计算，即子区间之间产生的贡献。
- 由于事先将元素按 a_i 大小排序。所以 $[mid+1,r]$ 中的元素的 a 值一定大于 $[l,mid]$ 中的元素。

「洛谷 P3810」陌上花开

- 我们将两个子区间中的数分别按照 b_i 排序。
- 利用归并排序的思路，按照 b_i 从小到大的顺序访问两个子区间中的数。
  - 如果该数在 $[l, mid]$ 中，即将其对应的 c_i 插入树状数组。
  - 如果该数在 $[mid+1, r]$ 中，即统计答案。
- 然后这个题做完了。
- 总时间复杂度为 $O(n\log_2^2 n)$。

根号分治

- 妙妙根号算法来啦！
- 广延各位的思路！！

「洛谷 P3396」哈希冲突

- 给定一个长度为 n 的序列，有 m 个操作：
  - 询问下标模 x 后为 y 的所有数之和
  - 修改第 x 个数

「洛谷 P3396」哈希冲突

- 思考暴力是怎么做的：
  - `for(int i=y;i<=n;i+=x) ans+=a[i];`
- 这个算法慢在哪里？
- 当 x 很小的时候很慢。
- 那么当 x 很小的时候有没有其他做法呢？
- 答案显然是有的。

「洛谷 P3396」哈希冲突

- 记 $a_{x,y}$ 为模 x 余 y 的答案。
- 当 x 较小时我们可以很快更新。
- 那么对于 x 较大或较小时我们都有非常优秀的做法。
- 那么以什么为分界线呢？
- 不难观察得到以 $\sqrt{n}$ 为分界线最优。
- 这也是根号分治的由来。

「洛谷 P6189」跑步

- 求 n 的自然数拆分方案数，对不定 p 取模。
- $n \leq 10^5$
- 提示：考虑 DP。

「洛谷 P6189」跑步

- 记 $f_{i,j}$ 表示当前使用了 $1\sim i$ 这些数字，总和为 j 的方案数。
- 由于 $1\sim i$ 这些数字都可以无限使用，所以这本质上就是完全背包。
- 若可使用的数字为 $1\sim k$，则此部分时间复杂度为 $O(nk)$。
- 那么这种解法在 k 较小时可行。

「洛谷 P6189」跑步

- 那么如果 k 较大呢？
- 注意到 n 的拆分最多只能包含 $\lfloor \frac{n}{k} \rfloor$ 个 k。
- 也就是说，实际上含有的大数是非常少的。
- 利用这个个数，我们可以设计一个 DP。
- 注意到如果我们强制要求自然数拆分数字有序其一定呈现阶梯状。
- 记 $g_{i,j}$ 为用 $\geq k$ 的 i 个数，总和为 j 的方案数。
- 那么有 $g_{i,j} = g_{i-1,j-k} + g_{i,j-i}$。
- 显然 $k = \sqrt{n}$ 非常优秀。
- 最后利用乘法原理组合即可。

题单 1

- 「NOIP2015」 跳石头
- 「NOIP2018」 赛道修建
- 「LOJ500」 ZQC 的拼图
- 「NOIP2015」 运输计划

- + 1580
- + 2601
- +6189
- + 1582

题单 2

- 「CF1385D」 a-Good String
- 「CF1442D」 Sum
- 「CF1609F」 Interesting Sections

- +5938
- +5939
- +5940

题单 3

- 「洛谷 P3810」陌上花开
- 「洛谷 P3396」哈希冲突
- 「洛谷 P6189」跑步
- 「UOJ246/FZOJ179」套路
- 还有更多更有意思的分治思想，等你从题目中探寻！

- + 3736
- + 5941
- +3601
- +5942