二分

crazy_cloud

SJTU

2021 年 9 月 26 日

目录

目录

这个专题讲什么?

二分，是计算机科学中最常用的算法之一，其思想其实也是非常贴近现实生活的。按照字母序在字典中找单词，或者是小学奥数会见到的找次品问题，都蕴含着二分的思想。
本专题将介绍二分的基础、应用和二分的一些衍生算法。前面的部分会比较基础，方便还不太了解这个算法的同学。

目录

二分查找算法

例 (经典题)

给出一个升序数列 $\{a_n\}$，现在有 m 个询问，每次询问比 x 大的最小的 a_i 的值。
$1 \leq n, m \leq 10^5$

假设对于下标 j，我们知道 $a_j \leq x$，这意味着什么？如果是 $a_j > x$ 呢？

二分查找算法

例 (经典题)

给出一个升序数列 $\{a_n\}$，现在有 m 个询问，每次询问比 x 大的最小的 a_i 的值。
$1 \leq n, m \leq 10^5$

当 $a_j \leq x$ 时，因为 $x < a_i$，所以 $j < i$。
当 $a_j > x$ 时，因为 a_i 是大于 x 的最小的数，于是 $i \leq j$。
如果我们把若干次查询下标 j 得到的信息综合起来，就能计算出 i 的可行区间 $[l, r]$。
为了使区间长度随着询问次数增加而快速缩小，我们选择 $mid = \lfloor (l + r)/2 \rfloor$ 作为每次查询的 j。这样，一次询问就能将区间长度减半。

二分查找的实现

Binary search

function BinarySearch
    $l = 1, r = n, ans = -1$
    while $l \leq r$ **do**
        $mid = \lfloor (l + r)/2 \rfloor$
        if $a[mid] \leq x$ **then**
            $l = mid + 1$
        else
            $r = mid - 1, ans = mid$
        end if
    end while
    return *ans*
end function

单次查找时间复杂度为 $O(\log n)$。

二分查找的条件

前面的例子可以发现，二分查找必须要在一个有序的数组中进行。因为只有数组有序，我们才可以通过当前值和目标值的大小关系，来得到目标值的可行区间。
单调性是贯彻所有二分算法的核心条件。

二分答案算法

二分查找往往是作为一些中间的小步骤出现，本身也很简单，因此我们不做太多的介绍。
二分答案是二分查找的一种具有更广泛运用场景的变体，他们的本质其实是一件事情。

例 (COCI 2011/2012 #5 - EKO)

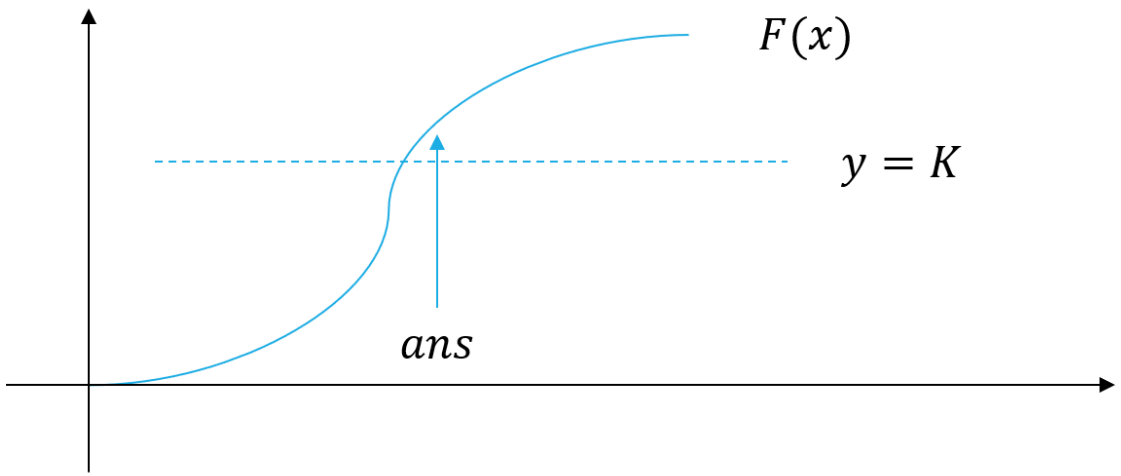
有 n 个非负整数 $a_1, \ldots, a_n$。
你需要设置一个参数 H，并获得

$$\sum_{i=1}^{n} \max\{0, a_i - H\}$$

的收益。
在收益至少为 m 的前提下，H 最大能是多少?
$1 \leq n \leq 10^6, 1 \leq m \leq 2 \times 10^9$

二分答案算法

如果 H_0 可行，那么小于 H_0 的所有 H 一定都可行，因为它们会带来更大的收益。
从小到大看，H 一开始不可行，在达到某个数值之后一直可行。
所以查找合适的 H 就好像是在升序序列中，寻找大于等于 m 的最小的数。
唯一不同的是，我们不再是简单地查询数组中的一个值，而是通过函数计算 H 为特定值时的收益。



二分答案算法

Binary search

function BinarySearch
  $l = 1, r = n, ans = -1$
  while $l \leq r$ **do**
    $mid = \lfloor (l + r)/2 \rfloor$
    if $f(mid) < m$ **then**
      $l = mid + 1$
    else
      $r = mid - 1, ans = mid$
    end if
  end while
  return *ans*
end function

单次查找时间复杂度为 $O(T(n)\log m)$。这里的 M 是搜索的值域，$T(n)$ 是计算函数 f 所需的时间。

例题

例 (NOIP 2015 - 跳石头)

长度为 L 的河道中，有 n 块石头，你需要移走至多 m 块，使得最短的间隔尽可能长（包括两端）。

$1 \leq n \leq 5 \times 10^4, 1 \leq L \leq 10^9$

例题

例 (NOIP 2015 - 跳石头)

长度为 L 的河道中，有 n 块石头，你需要移走至多 m 块，使得最短的间隔尽可能长（包括两端）。
$1 \leq n \leq 5 \times 10^4, 1 \leq L \leq 10^9$

可以发现，随着移动块数变多，答案单调不下降。这启发我们对答案进行二分。
我们令 $f(x)$ 表示当最小距离不低于 x 时，最少需要移动多少颗石头。二分找出最后一个 $f(x) \leq m$ 的 x 即为答案。
如何求 $f(x)$?

例题

例 (NOIP 2015 - 跳石头)

长度为 L 的河道中，有 n 块石头，你需要移走至多 m 块，使得最短的间隔尽可能长（包括两端）。
$1 \leq n \leq 5 \times 10^4, 1 \leq L \leq 10^9$

可以发现，随着移动块数变多，答案单调不下降。这启发我们对答案进行二分。
我们令 $f(x)$ 表示当最小距离不低于 x 时，最少需要移动多少颗石头。二分找出最后一个 $f(x) \leq m$ 的 x 即为答案。
如何求 $f(x)$?
一种贪心的策略是，从头开始，如果这一块石头与上一块石头的距离小于 x，就直接把这一块石头移去。最后统计移去的石头的总个数。这个策略显然是最优的。
时间复杂度 $O(n \log L)$。

例题

例 (NOIP 2012 - 借教室)

接下来的 n 天中，第 i 天有 r_i 个空教室可以租借。
共有 m 份订单，每份订单形如：从第 s_j 天到第 t_j 天都需要租借教室，每天需要租借 d_j 个教室。
订单按顺序处理，请计算第一个无法满足的订单。
$1 \leq n, m \leq 10^6$

例题

例 (NOIP 2012 - 借教室)

接下来的 n 天中，第 i 天有 r_i 个空教室可以租借。
共有 m 份订单，每份订单形如：从第 s_j 天到第 t_j 天都需要租借教室，每天需要租借 d_j 个教室。
订单按顺序处理，请计算第一个无法满足的订单。
$1 \leq n, m \leq 10^6$

对订单进行二分，现在问题变成给定若干个订单，问是否能满足。
其实就是有若干个区间加，然后最后询问是否有一个位置大于其阈值。

例题

例 (NOIP 2012 - 借教室)

接下来的 n 天中，第 i 天有 r_i 个空教室可以租借。
共有 m 份订单，每份订单形如：从第 s_j 天到第 t_j 天都需要租借教室，每天需要租借 d_j 个教室。
订单按顺序处理，请计算第一个无法满足的订单。
$1 \leq n, m \leq 10^6$

对订单进行二分，现在问题变成给定若干个订单，问是否能满足。
其实就是有若干个区间加，然后最后询问是否有一个位置大于其阈值。
由于这些区间加是给定的，我们可以差分处理，最后前缀和得到最终结果。
时间复杂度 $O((n+m)\log m)$。

例题

例 (SYCOJ #319 - 团队协作)

有 n 个人，每个人有个权值 a_i。
你需要把 n 个人分成若干组，每组大小至少为 k，请最小化组内权值极差的最大值。
$1 \leq n \leq 3 \times 10^5$

例题

例 (SYCOJ #319 - 团队协作)

有 n 个人，每个人有个权值 a_i。
你需要把 n 个人分成若干组，每组大小至少为 k，请最小化组内权值极差的最大值。
$1 \leq n \leq 3 \times 10^5$

二分答案，难点在于如何 check。
将所有人按照权值排序，一定存在一种最优分组方案满足每个小组是连续的。我们从前往后计算，设 f_i 表示只考虑前 i 个人，是否有一种方案恰好将这 i 个人分好且极差满足条件。
可以发现转移的可行区间左右端点各自单调，用 two pointers 即可做到线性。
时间复杂度 $O(n \log n)$。

例题

例 (Monument)

有 n 个人在一条无穷的数轴上，一开始第 i 个人在 p_i。他们从零时刻开始以每秒 v_i 的速度移动。
现在你最多可以删除 K 个人，最大化 T 使得在前 T 秒内，不存在任意两个人曾经相遇（位置相同即相遇，包括追及）。
$1 \leq n \leq 10^5, 0 \leq |p_i|, |v_i| \leq 10^9$

例题

例 (Monument)

有 n 个人在一条无穷的数轴上，一开始第 i 个人在 p_i。他们从零时刻开始以每秒 v_i 的速度移动。
现在你最多可以删除 K 个人，最大化 T 使得在前 T 秒内，不存在任意两个人曾经相遇（位置相同即相遇，包括追及）。
$1 \le n \le 10^5, 0 \le |p_i|, |v_i| \le 10^9$

对时间二分答案。
那么怎么判定答案 T 是否可行呢？可以发现，两个人没有相遇当且仅当在零时刻两人的相对顺序和 T 时刻一致。
那么我们删掉最少的人等价于保留最多的人使得这些相对顺序不发生变化。
那可以一开始将所有人按照 p_i 排序，然后判定时计算每个人的位置，做一个最长上升子序列就可以得到最多能保留的人数。
时间复杂度 $O(n \log n \log T)$。

总结：二分的套路

首先观察单调性，找到需要计算的条件。
接下来就可以二分，并验证条件了。
二分答案的两个难点：找单调性和验证。
通常，最小化某个最大值或者最大化某个最小值这样的题目要求，都可以考虑使用二分。

目录

最长不下降子序列

最长不下降子序列问题 (LIS)

如果一个序列 $\{a_n\}$ 满足 $a_1 \leq a_2 \leq \cdots \leq a_n$，我们称其为不下降序列。
特殊地，当这个序列的数互不相等时，我们也称其是递增序列。
同理，还有不上升序列和递减序列。
给定一个长度为 n 的序列，求出它最长不下降子序列长度。

最长不下降子序列

假设现在我们考虑到下标 i，已知前面已经有若干个长度为 l 的不下降子序列。
我需要保存哪些信息，才能让后来的元素加入形成长度为 $l+1$ 的不下降子序列这种情况被计算入内？
这若干个不下降子序列具体的组成吗？我并不关心，我只需要最后一个数的大小就能用以转移。
这若干个不下降子序列最后一个数都需要吗？并不是，显然我们只关心最小值，比它大的子序列都是无用的，因为只要能在这些序列后面接上 x，那么这个包含着末尾最小值的子序列一定也能接上 x。这是 LIS 的贪心性质。

最长不下降子序列

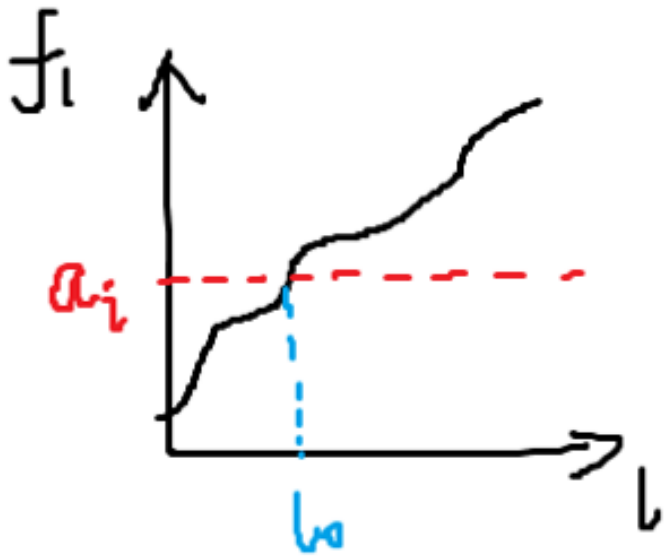
这启发了我们保留每个长度不下降子序列的最小末尾。
设 f_l 表示只考虑我们现在计算到的前缀，所有长度为 l 的子序列，末尾最小值。如果不存在长度为 l 的子序列，则设置为无穷大。
从前往后枚举原序列，不断维护 f_l。
现在我们要加入 a_i，那么前面所有末尾小于等于 a_i 的子序列都可以接上 a_i 形成长度增加 1 的不下降子序列。
枚举所有 l，如果 $f_l < a_i$，那么我们就能在其后面接上 a_i 形成长度为 $l+1$ 的不下降子序列。
于是 $f_{l+1} = \min\{a_i, f_{l+1}\}$。
可是这样是 $O(n^2)$ 的。

最长不下降子序列

观察 f_l 的性质。可以发现 f_l 单调不下降!
因为如果存在长度为 l 的子序列以 a 结尾，则截取后 $l-1$ 位就可以得到长度为 $l-1$ 的以 a 结尾的子序列。
因此，随着位数的增大，这个最优结尾大小只会变大。

最长不下降子序列

对于 a_i，我们设 l_0 是最大的满足 $f_l \leq a_i$ 的 l。
于是，对于所有大于 l_0 的 l，其子序列结尾最大值都要大于 a_i，不能接上 a_i 成为更长的子序列。
对于所有小于 l_0 的 l，f_l 虽然能接上 a_i，但是 $a_i \geq f_{l+1}$，并不能使得末尾变小。
于是只有 $f_{l_0+1} = a_i$。
二分找 l_0，时间复杂度 $O(n \log n)$。



分数规划问题

分数规划问题

有 n 个物品，每个物品有两个权值 a 和 b。求一组 $w_i \in \{0, 1\}$，最大化

$$\frac{\sum a_i \times w_i}{\sum b_i \times w_i}$$

的值。

即选择一些物品使得它们 a 的总和与 b 的总和比值尽量大。

分数规划问题

如果最优解比 x 大意味着什么？
存在 w 满足

$$\frac{\sum a_i \times w_i}{\sum b_i \times w_i} > x$$

$$\sum a_i \times w_i - x \sum b_i \times w_i > 0$$

$$\sum (a_i - b_i x) \times w_i > 0$$

现在我们可以将每个物品的权值看成 $a_i - b_i x$，然后判断是否存在一种选取方案满足权值和大于零。
显然我们只需要判断是否存在正的权值就好。
有了判定方法，我们可以对 x 进行二分，找到最优解。

例题

例 (Luogu 4377 - Talent Show)

有 n 个物品，每个物品有两个权值 a 和 b。求一组 $w_i \in \{0, 1\}$，最大化

$$\frac{\sum a_i \times w_i}{\sum b_i \times w_i}$$

的值。
要求 $\sum w_i \times b_i \geq W$。

例题

例 (Luogu 4377 - Talent Show)

有 n 个物品，每个物品有两个权值 a 和 b。求一组 $w_i \in \{0, 1\}$，最大化

$$\frac{\sum a_i \times w_i}{\sum b_i \times w_i}$$

的值。
要求 $\sum w_i \times b_i \geq W$。

做法基本上和前面基本一致，分母大小限制可以通过将判定问题转化为 $0/1$ 背包解决。