二分、倍增

二分法

- 二分法本质上永远基于单调性
- 通过使用二分法可以把最优化问题转化为判定性问题
- 二分判断并不只限于“比目标值大/比目标值小”，只要能判断出目标值在哪边都行——排除一半状态
- 经典关键词：“最大值最小”、“最小值最大”

  ——最小权值最大的完全匹配、瓶颈生成树、权值波动最小的路径

序列上二分

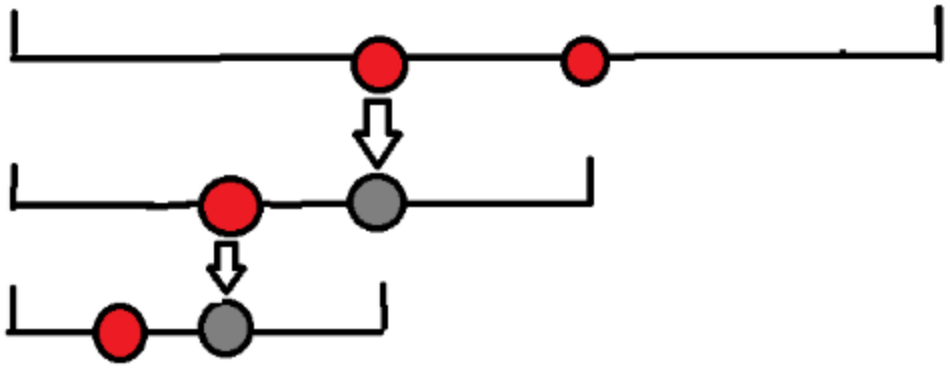
- 在从小到大的排序数组中：
- lower_bound(begin,end,num)：左闭右开，二分查找第一个大于或等于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。
- upper_bound(begin,end,num)：左闭右开，二分查找第一个大于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。

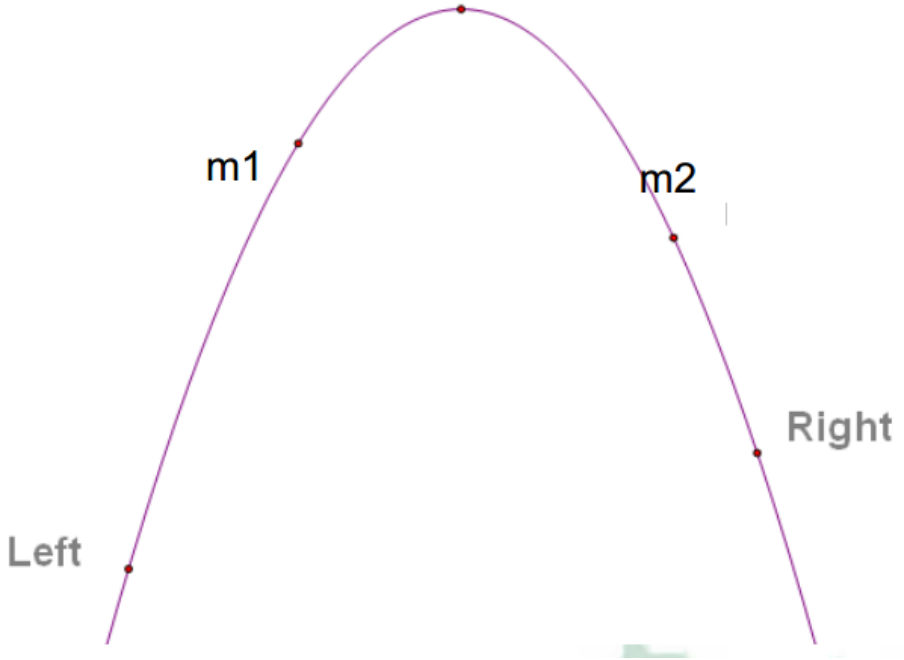
实数域上二分

- 固定精度： Epsilon
- 固定次数：更高精度

三分法

- 单峰函数极值：取与极值较“远”的点，丢弃外侧的1/3
- 黄金分割法：减少取点操作。





例：分段（循环）有序队列

- 一个数列，它可以分为前后两个部分，两段各是一个递增数列，并且后一段的最大值比前一段的最小值还要小，如何在这个数列中查询指定的元素。

- 和端点比较，找拐点

例：k小数

- 两个有序数组，如何找出其中第k小的数

- 两个数组都取中点，比较两个数组前半段的**长度和**与k的大小

例

- n个数的有序序列，这n个数两两的差值将产生n(n-1)/2个数。试求这n(n-1)/2个数的中位数。

- 二分答案加二分检测。先二分这个中位数，然后数一数比该数小的有多少，以确定这个“中位数”是取大了还是取小了。为了判断有多少个数比选定的中位数小，再一次使用二分：对于每一个固定的数A_i，二分A_j的位置，看A_i-A_j比中位数大还是小（j=1..i-1）。假设n个数中的最大值为q，则算法的复杂度为O(n*log(n)*log(q))。

分数规划

分数规划用来求一个分式的极值。

形象一点就是，给出 a_i 和 b_i ，求一组 $w_i \in \{0,1\}$ ，最小化或最大化

$$\frac{\sum_{i=1}^{n} a_i \times w_i}{\sum_{i=1}^{n} b_i \times w_i}$$

另外一种描述：每种物品有两个权值 a 和 b ，选出若干个物品使得 $\frac{\sum a}{\sum b}$ 最小/最大。

一般分数规划问题还会有一些奇怪的限制，比如『分母至少为 W 』。

分数规划问题的通用方法是二分。

假设我们要求最大值。二分一个答案 mid，然后推式子（为了方便少写了上下界）：

$$\begin{aligned}
&\frac{\sum a_i \times w_i}{\sum b_i \times w_i} > mid \\
\Longrightarrow &\sum a_i \times w_i - mid \times \sum b_i \cdot w_i > 0 \\
\Longrightarrow &\sum w_i \times (a_i - mid \times b_i) > 0
\end{aligned}$$

那么只要求出不等号左边的式子的最大值就行了。如果最大值比 0 要大，说明 mid 是可行的，否则不可行。

倍增

- 基础：二进制拆分

1. 处理没有上界（或很大）的二分
2. 按位检验 2 的整次幂的累加和，逼近答案。例：Genius ACM（也有叫倍增二分的）
3. 利用二进制，预处理出以 2 的整次幂为单位的信息。（ST表）

- 结合：倍增求LCA（预处理祖先节点，从高位到低位尝试找到LCA）。

例题：我也不是B

- 开始有一个空序列s,一个变量c=0，接着从左往右依次将数组a中的数字放入s的尾部，每放一个数字就检测一次混乱度K，当混乱度k大于M时就清空序列并让c=c+1
- K = Bi * Vi(1<=i<=k(序列总长度)的总和)，Bi表示序列中第i小的数字，Vi是给定的非递减的数，输出每次加入序列后的变量c

- 首先发现当没有清空的时候每次向后增加K都不会减小，所以对于不清空来说K一定是非递减的
- 定义左端点L二分寻找每个右端点R，保证现在R是[L,R]第一个大于M的位置，寻找内部是直接排序暴力
- 但是会超时，因为可能每个R都离L不远，这样每次减去的数字就很少
- 我们考虑另一种二分，首先根据L暴力找到p，使[L,2^P]是第一个大于M的位置，这儿最多是暴力logn次，而且当2^p不是很大时，**每次暴力内部都很快（[L,2^p]数字少）**
- 接着二分寻找[L,2^(p-1)]与[L,2^p]中最小的R满足[L,R]第一个大于M，这儿减去的数字少的话也很快，减去的多等等循环的次数就少了

例：疫情控制（NOIP2012D2T3）

题目描述

H 国有 n 个城市，这 n 个城市用 $n-1$ 条双向道路相互连通构成一棵树， 1 号城市是首都，也是树中的根节点。

H 国的首都爆发了一种危害性极高的传染病。当局为了控制疫情，不让疫情扩散到边境城市（叶子节点所表示的城市），决定动用军队在一些城市建立检查点，使得从首都到边境城市的每一条路径上都至少有一个检查点，边境城市也可以建立检查点。但特别要注意的是，首都是不能建立检查点的。

现在，在 H 国的一些城市中已经驻扎有军队，且一个城市可以驻扎多个军队。一支军队可以在有道路连接的城市间移动，并在除首都以外的任意一个城市建立检查点，且只能在一个城市建立检查点。一支军队经过一条道路从一个城市移动到另一个城市所需要的时间等于道路的长度（单位：小时）。

请问最少需要多少个小时才能控制疫情。注意：不同的军队可以同时移动。

二分时间

然后一个显然的事是一个军队向上爬的越高它控制的点越多

所以首先军队尽量往上爬。

当一个军队可以爬到根节点我们记录下它的剩余时间T和它到达根结点时经过的根节点的子节点son。

当一个军队爬不到根节点时我们就让它控制它可以爬到的最高点。

然后我们把爬到根节点的军队按T从小到大排序。

然后按顺序处理

然后假如一个军队没有时间回到它的son，且son还没有控制。就让它控制son。

因为让别的军队去控制它显然更浪费时间。

否则贪心地匹配就好了（尽量小的和小的和匹配）

然后向上爬用倍增来优化。

例：软件开发

- 一个软件开发公司同时要开发两个软件，并且要同时交付给用户，现在公司为了尽快完成这一任务，将每个软件划分成m个模块，由公司里的技术人员分工完成，每个技术人员完成同一软件的不同模块的所用的天数是相同的，并且是已知的，但完成不同软件的一个模块的时间是不同的，每个技术人员在同一时刻只能做一个模块，一个模块只能由一个人独立完成而不能由多人协同完成。一个技术人员在整个开发期内完成一个模块以后可以接着做任一软件的任一模块。写一个程序，求出公司最早能在什么时候交付软件。 1≤n≤100,1≤m≤100

- 求所有人完成时间最长的最小值-->最大/小 化 最/大小值-->二分答案
- 如何验证在时间lim的范围之内是否能完成工作?
- 我们发现由于同一个人分配在1、2上面的时间不同，会对结果产生影响，所以我们要**解决两个工作之间的干扰问题**
- 由于n,m不大，试着枚举前i个人在总共做了j项工作1时，剩下还可以做工作2份数的最大值，设为dp[i][j]
- 枚举第i个人做了k份工作i，则状态转移方程为:
- dp[i][j]=max{ dp[i][j-k] + (lim-a[i]*k) / b[i] }
- 最后如果dp[n][m]<m，则不能完成，反之则可以;

Others:

- 二分+2-SAT、二分加网络流、线段树+二分
- 倍增优化动态规划
- 倍增floyd、倍增FFT等与其他算法的结合
- 后缀数组等算法也运用了倍增思想
- Chan凸包算法