[13] 二分

深入浅出程序设计竞赛

第 2 部分 – 初涉算法

V 2022-06

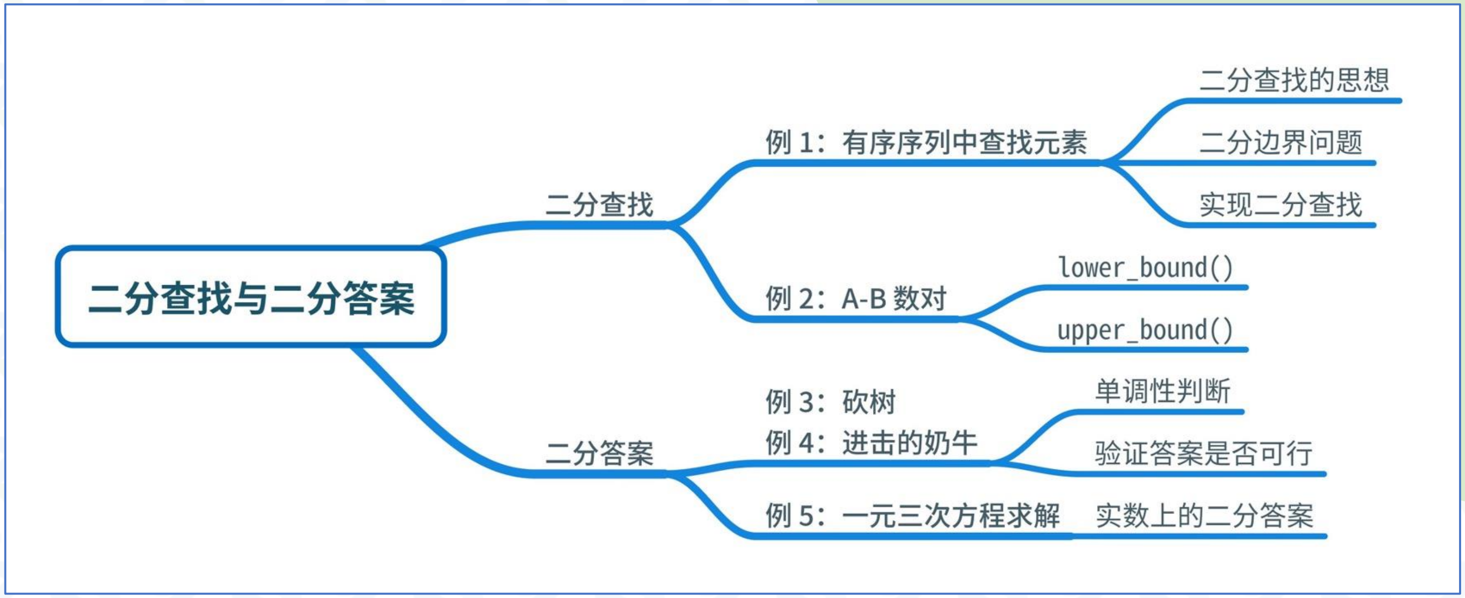
洛谷

www.luogu.com.cn

版权声明

- 其它《深基》配套资源、购买本书等请参阅：https://www.luogu.com.cn/blog/kkksc03/IPC-resources
- 如果课件有任何错误，请在这里反馈 https://www.luogu.com.cn/discuss/show/296741

本章知识导图



二分查找与二分答案
二分查找
例 1：有序序列中查找元素
二分查找的思想
二分边界问题
实现二分查找
例 2：A-B 数对
lower_bound()
upper_bound()
二分答案
例 3：砍树
例 4：进击的奶牛
单调性判断
验证答案是否可行
例 5：一元三次方程求解
实数上的二分答案

第 13 章 二分法

二分查找

让元素查找变得更高效!

请翻至课本 P174

猜数游戏

- 电脑随机生成一个 1 到 100 的整数。
- 你可以进行多次猜测。

- 对于每次猜测，你需要输入一个数字。
- 如果你输入的数字和计算机生成的一致，则成功。
- 如果你输入的数字比计算机生成的更大，告诉你更大。
- 如果你输入的数字比计算机生成的更小，告诉你更小。

多少次一定可以才出来呢？

演示一下这个游戏！

猜数游戏

策略 1:

从 1 开始猜测，然后猜 2……，最后猜 100。

- 运气好的时候（随机数是 1），一次猜中。
- 运气不好的话（随机数是 100），要猜 100 次。

如果随机产生的数字，平均大约猜 50 次可以猜中。

那效率有点低。

猜数游戏

策略 2:

- 先猜测 1 和 100 中间的数字 50。
- 如果运气不错，刚好就是 50，那么只需猜一次。
- 如果 50 太小了，则范围缩到 51 到 100。
- 如果 50 太大了，则范围缩到 1 到 49。

猜数游戏

策略 2（继续）：

- 剩下 49 个数字，继续猜范围中间的数字。
- 假设范围是缩到 51 到 100。我们猜测中间数 75。
- 如果 75 太小，则把范围缩到 51 到 74。
- 如果 75 太大，则把范围缩到 76 到 100。

猜数游戏

策略 2（再继续）：

- 假设最后剩下两个数字 69 到 70。我们猜测中间数 69。
- 如果 69 正确，答案就是 69
- 如果 69 太大，那么无解（不可能发生）
- 如果 69 太小，则把范围缩到 70 到 70。

这样，就把范围限制在长度为 1 的区间中，直接找到答案。

猜数游戏

策略:

- 在指定的区间，尝试中间值。
- 如果中间值就是答案则输出答案。
- 如果中间值太小，则继续处理右区间。
- 如果中间值太大，则继续处理左区间。
- 每次都可以把可能的数字缩小一半。

  100→50→25→12→6→3→1
- 因此，最多只需要 7 次就可以找到答案。

这种每次淘汰掉一半区间，最后只留下一个的做法，是二分查找。

查找

例 12.1 （洛谷 P2249）

输入 $n(n \leq 10^6)$ 个不超过 10^9 的单调不减的（就是后面的数字不小于前面的数字）非负整数 $a_1, a_2, \dots, a_n$

然后进行 $m(m \leq 10^5)$ 次询问。对于每次询问，给出一个整数 $q(q \leq 10^9)$，要求输出这个数字在序列中的编号。

如果没有找到的话输出 -1 。

样例输入:

```
11 3
1 3 3 3 5 7 9 11 13 15 15
1 3 6
```

样例输出:

```
1 2 -1
```

查找

直接从头到尾搜索一遍查找数字是不可行的。如果查找数字太多的话复杂度就是 $O(mn)$，运行效率太低。

考虑从中间的数字开始查找。

- 若序列中最中间的数字等于要找的数字：直接返回即可
- 这个数字小于要找的数字：由于序列升序，因此序列的前半段显然都小于我们寻找的目标。
  因此，只需要从序列的后半段寻找即可。
- 这个数字大于要找的数字：同理，只要从前半段寻找即可。

查找

- 序列中间等于要找的数：直接返回；
- 序列中间大于要找的数：继续找前半段；
- 序列中间小于要找的数：继续找后半段；

```
int find(int x) {
    int l = 1, r = n;
    while (l <= r) {
        int mid = (l + r) / 2; // 中间页数
        if (a[mid] == x)return mid; // 刚好找到需要的数字
        else if (a[mid] > x)r = mid - 1; // 取区间的前一半
        else l = mid + 1; // 取区间的后一半
    }
    return -1; // 最后没有找到
}
```

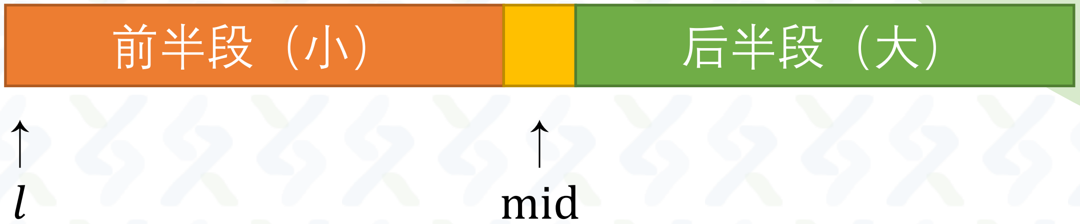
问题：比如遇到重复的数字，就会直接输出最先找到的编号。

解决：可先通过二分法找出序列中第一个大于等于 q 的数，再判断它是否为 q，即可得到最终答案。

查找

二分过程中，如果 mid 位置的值大于等于 q，就将 r 指针移动到 mid，否则就将 l 指针移动到 mid+1。

这样 r 永远是大于等于 q 的位置，而 $l-1$ 永远是小于 q 的位置。而当两者相等时，就是分界线的位置。



然而事实上，由于 q 可能本身大于整个序列中最大的数，因此我们需要将 r 的初始值设置为 n+1，才能保证程序的正确运行。

查找

第1轮
l=1
mid=6
r=11
1 2 3 4 5 6 7 8 9 10 11
1 3 3 3 5 7 9 11 13 15 15
a[mid]>=3成立（大了）
取左区间
第2轮
l=1
mid=3
r=6
1 2 3 4 5 6 7 8 9 10 11
1 3 3 3 5 7 9 11 13 15 15
a[mid]>=3成立（大了）
取左区间
第3轮
l=1
mid=2
r=3
1 2 3 4 5 6 7 8 9 10 11
1 3 3 3 5 7 9 11 13 15 15
a[mid]>=3成立（大了）
取左区间
第4轮
l=mid=1
r=2
1 2 3 4 5 6 7 8 9 10 11
1 3 3 3 5 7 9 11 13 15 15
a[mid]>=3不成立（小了）
取右区间
最终
l=2，r=2，退出循环

查找

在找到了满足条件的数后，再与 q 进行比较并返回结果，即可完成查询。代码如下：

```
int find(int x) {
    int l = 1, r = n + 1;
    while (l < r) { // 最后 l 和 r 会相等。
        int mid = l + (r - l)/2;
        // 有时 l+r 可能会超过 int 类型的极限（当然本例不会），这么做可以避免运算溢出。
        if (a[mid] >= x)r = mid;
        else l = mid + 1;
    }
    if(a[l] == x)return l;
    else return -1;
}
```

由于每轮二分区间长度都要衰减一半，因此二分查找的复杂度是 $O(\log n)$，相比于直接枚举搜索的 $O(n)$ 有了很大的改进。

查找

其实在STL里面也有已经实现好的二分函数，包含于algorithm库中。

- lower_bound(begin,end,val)：在有序数组 [begin,end) 中找到第一个大于等于 val 的值并返回其地址。
- upper_bound(begin,end,val)：在有序数组[begin,end)中找到第一个大于 val 的值并返回其地址。

和大多数 STL 函数一样，begin 指数组首地址，end 指末地址+1。

查找

需要注意的是，如果在数组中返回得到的结果将会是一个地址，而我们经常需要将其转化为下标。

例如下面的代码得到的就是 a 数组（下标从0开始）第一个大于等于 val 的值的下标。

```
int pos = lower_bound(a, a+n, val) - a;
```

二分答案

将求解化为判断，让问题更简单！

请翻至课本 P179

快速回顾

之前讲过猜数游戏。简而言之:

- 随机生成一个 100 以内的正整数
- 玩家每次选择一个 x，电脑会告知答案是比 x 大、相等还是比 x 小。

通过合适的策略选择，最多用 $O(\log n)$ 次询问，找到答案。

基本思路：每次把「解的搜索空间」排除一半。

二分答案

二分思想不仅可以在有序序列中快速查询元素，还能高效率地解决一些具有单调性判定的问题。

事实上，二分查找的本质，就是在寻找一个满足“a[mid]>=k”这一条件的最小的 mid。由于 a 数组的单调性，我们设计出了这一算法。

事实上，在一些具有单调性判定的问题中，我们也可以采用类似方法求解。

砍树

例 3：（洛谷 P1873）

N 棵树高度分别为$a_1 \dots a_N$，对于一个砍树高度 H，可以锯下并收集到每棵树上比 H 高的部分的木材。求最大的整数高度 H，使得能够收集到长度为 M 的木材。其中 $N \le 10^6$，树高不超过 10^9。

样例输入：

```
5 20
4 42 40 26 46
```

样例输出：

```
36
```

在样例中，每棵树的高度分别是 [4,42,40,26,46]，需要将锯子的高度调整为 36，这样可以分别锯下 [6,4,10] 高度的木材。如果锯子高度再高一点就不能满足要求了。

砍树

先来变换一下题目：令“条件”表示“当砍树高度为 x 时可以获取不少于 M 的木材”，那么就是要找最大的 x 使得“条件”成立。

显然，这一条件是存在单调性的。斧头高度越低，则砍下时，收获的木头越多。即然如此，就可以采用二分的思路来寻找这个数。

```
int Find(int L, int R) {//使用前确保答案在[L,R]内。
    int ans, mid;
    while (L <= R) {//闭区间上的二分结束条件
        int mid = L + R >> 1;
        if (P(mid)) // 条件成立
            ans = mid, r = mid - 1;
        //这里只需要记录满足条件的mid，最后循环一定会结束也一
定会在ans中保留正确的答案
        else
            l = mid + 1;//l和r不用仔细考虑加1减1，全都写上去。
    }
    return ans;//其实ans=r-1，想一想为什么
}
```

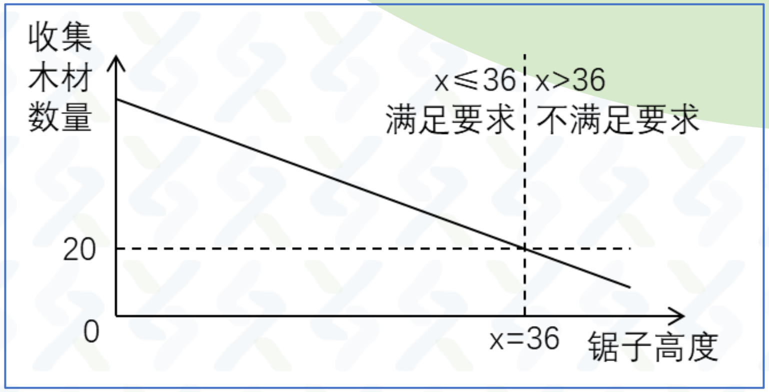
砍树

接下来，我们的问题就改变为如何判断给定的 x 是否能使条件成立（即 P 函数）。

当砍树高度为 x 时，对于某棵树，如果其高度高于 x，则砍去高于 x 的部分，否则舍弃，就能统计处砍掉的高度。

接着将砍去的高度求和，再与 M 相比即可。

```
bool P(int h) { // 当砍树高度为h时能否得到大于m的木材
    LL tot = 0;
    for (int i = 1; i <= n; i++)
        if (a[i] > h)
            tot += a[i] - h; // 按照题意模拟。
    return tot >= m;
}
```



砍树

需要注意的是，这里的 R 的初始值应当大于等于所有树中最高者的高度，否则可能无法通过二分的到正确的结果。

需要注意的是，这里的 R 的初始值应当大于等于所有树中最高者的高度，否则可能无法通过二分的到正确的结果。

判断条件是否成立的算法复杂度是 $O(n)$，而二分答案本身的算法度复杂度是 $O(\log A)$，其中 A 是指最高的高度，因此总复杂度是 $O(n\log A)$。

判断条件是否成立的算法复杂度是 $O(n)$，而二分答案本身的算法度复杂度是 $O(\log A)$，其中 A 是指最高的高度，因此总复杂度是 $O(n\log A)$。

砍树

通过本题， 我们可以得到使用二分答案的条件:

1. 命题可以被归纳为找到使得某命题P(x)成立（或不成立）的最大（或最小）的x。
2. 把P(x)看做一个值为真或假的函数，那么它一定在某个分界线的一侧全为真，另一侧全为假。
3. 可以找到一个复杂度优秀的算法来检验P(x)的真假。

通俗来讲，二分答案可以用来处理“最大的最小”或“最小的最大”问题。

进击的奶牛

例 4： （P1824，USACO 未知年份比赛）

一个牛棚有 N 个隔间，它们分布在一条直线上，坐标是 $x\_1,\ldots,x\_N$ 。现在需要把 C 头牛安置某些隔间，使得所有牛中相邻两头的最近距离越大越好，求这个最大的最近距离。

样例输入：

```
5 3
1
2
8
4
9
```

样例输出：

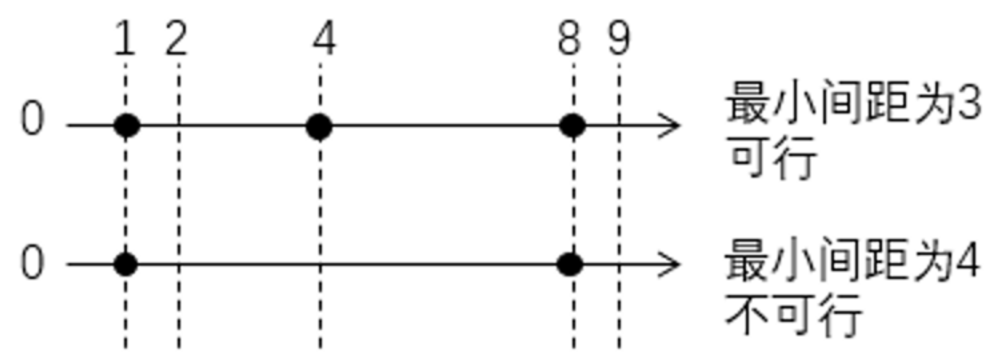
```
3
```

有 5 个隔间，3 头牛，隔间的坐标是 [1,2,8,4,9]。可以将牛关在 [1,4,9] 这些隔间中，最近的距离是 3。如果要求所有牛之间距离大于 3，是办不到的。

进击的奶牛

按照套路可以构造判断“条件”：可以把 C 头牛全部安置进这些隔间使相邻两头牛距离不超过 x。

不难发现，x 越大则条件越难满足。因此存在一个分界线 ans，x 大于 ans 时不存在合法安置方案，x 小于等于 ans 时则存在。



显然可以采用二分的思路。接下来我们来考虑如何判断某个间距是否可行。

进击的奶牛

本题只有一个限制，即任意两个相邻安置点距离不能小于 x。于是可以大致感受到一种贪心算法：从最左端开始遍历所有安置点，能安置就安置。

容易证明，在本题中，安置一定比不安置更优，贪心正确。因此，我们只需要遍历所有点并记录总共安置了多少头牛，再判断是否多余需安置的数量即可。

```
bool P(int d) {
    int k = 0, last = -INF; // last记录上一头牛的安置坐标
    for (int i = 1; i <= n; i++)
        if (a[i] - last >= d) // 能安置就立刻安置
            last = a[i], k++;
    return k >= c;
}
```

一元三次方程求解

例 5：（洛谷 P1024，NOIP2001 提高组）

解方程 $ax^3+bx^2+cx+d=0$，保证有三个实数根，且都在[-100,100]上，还保证任意两根之差不小于1。要求精确到小数点后 2 位。

样例输入：

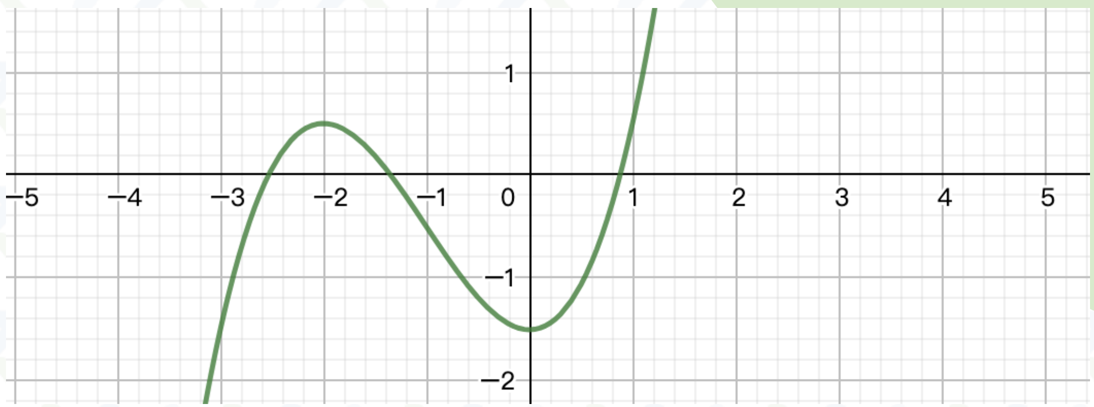
```
1 -5 -4 20
```

样例输出：

```
-2.00 2.00 5.00
```

一元三次方程求解

在解题之前，引出零点存在性定理：对连续函数 f(x) 若有 f(a)×f(b)<0 (a<b)，则 f(x) 在区间 (a,b) 上至少存在一个解。这样就可以判断一个区间中是否存在解。

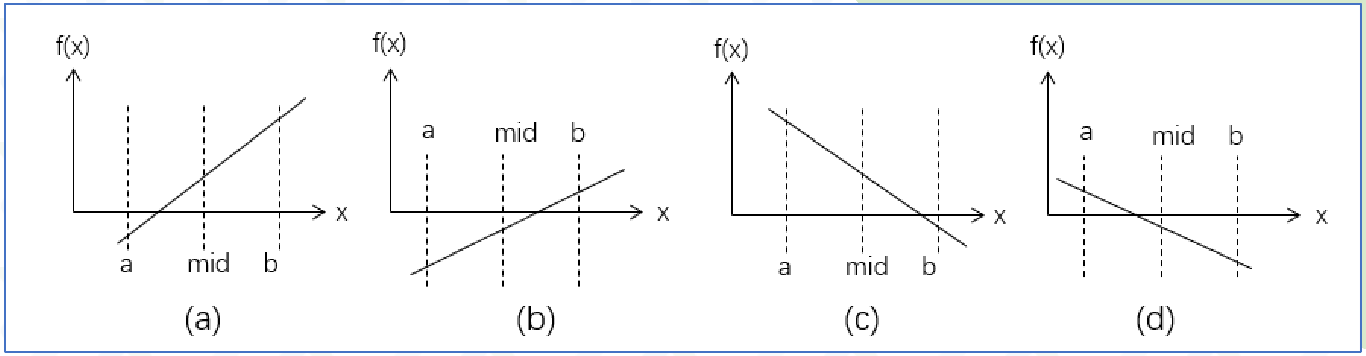


题目里说明了任意两根之差不小于 1，那么可以把 [-100,100] 等分成若干小段差距为 1 的若干小段。若一段两边正负号不同，则代表其中有根。

一元三次方程求解

接着，我们通过二分的方式寻找零点：如果中点的函数值和某端点的正负性相同，那么零点一定在中点的另一侧。

具体的，我们可以分为如下四种情况。



而判断两数符号是否相同，则可以转化为判断两数乘积的正负。故上图可以等价于

```
if (f(mid) * f(R) > 0) R = mid;
else L = mid;
```

一元三次方程求解

因此可以完成这样的程序。注意实数之间不能直接比较是否相等，而是判断之间的差值是否小于一个极小值。最终代码如下：

```
for (int i = -100; i <= 100; i++) {
    double L = i, R = i + 1, mid; //这里只处理区间[L,R)上的根
    if (fabs(f(L)) < eps)//如果L是根，可以直接输出
        printf("%.2lf ", L);
    else if (fabs(f(R)) < eps)//如果R是根，跳过
        continue;
    else if (f(L) * f(R) < 0) {//在(L,R)上有根，执行二分
        while (R - L > eps) {
            mid = (L + R) / 2;
            if (f(mid) * f(R) > 0)
                R = mid;//如果f(mid)和f(R)正负性相同，那么零点在mid左侧
            else
                L = mid;//否则在另一侧
        }
        printf("%.2lf ", L);
    }
}
```

课后习题与实验

学而时习之，不亦说乎。学而不思则罔，思而不学则殆。——孔子

请翻至课本 P184

复习

二分查找 在有序数列中，每次判断中间项与目标向的大小，lower_bound 与 upper_bound。

二分答案 通过对有单调性的问题可能的解的范围进行二分，得出满足条件的分界线。

作业

习题 13.1：跳石头（P2678，NOIP2015 提高组）

“跳石头”比赛将在一条笔直的河道中进行，河道中分布着一些巨大岩石，其中两块岩石作为比赛起点和终点，终点距离起点为 $L(L\leqslant10^9)$。

在起点和终点之间，有 $N(N\leqslant50000)$ 块岩石（不含起点和终点），距离起点的距离为 $D_i(0<D_i<L)$。

选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

主办方计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。至多从起点和终点之间移走 M 块岩石（不能移走起点和终点的岩石）。

求最短跳跃距离的最大值。

作业

习题 13.2 烦恼的高考志愿（洛谷 P1678）

现有 $m(m \le 100000)$ 所学校，每所学校预计分数线是 $a_i(a_i \le 10^6)$。有 $n(n \le 100000)$ 位学生，估分分别为 $b_i(b_i \le 10^6)$。

根据预计分数线和学生的估分情况，分别给每位学生推荐一所学校，要求学校的预计分数线和学生的估分相差最小（可高可低），这个最小值为不满意度。求所有学生不满意度和的最小值。

习题 13.3 木材加工（洛谷 P2440）

有一些原木，割成一些长度相同的小段木头（可能有剩余），需要得到的小段的数目是给定的。我们希望得到的小段木头越长越好，你的任务是计算能够得到的小段木头的最大长度。原木的长度都是正整数，要求切割得到的小段木头的长度也是正整数。

作业

习题 13.4 路标设置（洛谷 P3853，天津市队选拔 2007）

B 市和 T 市之间有一条 $L(L \le 10^7)$ 公里高速公路，这条公路的 $N(N \le 100000)$ 个点上设有路标。我们把公路上相邻路标的最大距离定义为该公路的“空旷指数”。现在政府决定在公路整数公里点上增设 $K(K \le 100000)$ 个路标，使得公路的“空旷指数”最小。他们请求你设计一个程序计算能达到的最小值是多少。请注意，公路的起点和终点保证已设有路标，公路的长度为整数，并且原有路标和新设路标都必须距起点整数个单位距离。

习题 13.5 银行贷款（洛谷 P1163）

当一个人从银行贷款后，在一段时间内这人将不得不每月偿还固定的分期付款。已知贷款的本金 P，每月支付的分期付款金额 A，分期付款还清贷款所需的总月数 M。这个问题要求计算出贷款者向银行支付的月利率 ans，按百分比输出。

作业

习题 13.6 数列分段 - Section II（洛谷 P1182）

对于给定的一个长度为 $N(N \leq 100000)$ 的正整数数列$A_i(A_i \leq 10^9)$，现要将其分成$M(M \leq N)$段，并要求每段连续，且每段和的最大值最小。例如一数列 [4,2,4,5,1] 要分成 3 段，将其分成 [4,2] [4,5] [1] 时，第一段和为 6，第 2 段和为 9，第 3 段和为 1，和最大值为 9；将其分成 [4] [2,4] [5,1] 时，第一段和为 4，第 2 段和为 6，第 3 段和为 6，和最大值为 6；无论如何分段，最大值不会小于 6。求每段和最大值最小为多少。

作业

习题 13.7 （选做）kotori的设备（洛谷 P3743）

kotori 有 $n(n \le 100000)$ 个可同时使用的设备。第 i 个设备平均每秒均匀消耗 $a_i(a_i \le 10^6)$ 个单位能量（每时每刻都在消耗，并不是每一秒钟一下子扣除 a_i 能量哦）。在开始的时候第 i 个设备里存储着 $b_i(b_i \le 10^6)$ 个单位能量。同时 kotori 又有一个可以给任意一个设备充电的充电宝，平均每秒可以给接通的设备充能 $p(p \le 10^6)$ 个单位，充能也是连续的。你可以在任意时间给任意一个设备充能，从一个设备切换到另一个设备的时间忽略不计。kotori 想把这些设备一起使用，直到其中有设备能量降为 0。所以 kotori 想知道，在充电器的作用下，她最多能将这些设备一起使用多久。至少精确到 6 位小数。