

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 』。

求解

二分法

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）：

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

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。

求最小值的方法和求最大值的方法类似，读者不妨尝试着自己推一下。

Dinkelbach 算法

Dinkelbach 算法的大概思想是每次用上一轮的答案当做新的 L 来输入，不断地迭代，直至答案收敛。

分数规划的主要难点就在于如何求 $\sum w_i \times (a_i - mid \times b_i)$ 的最大值/最小值。下面通过一系列实例来讲解该式子的最大值/最小值的求法。

实例

模板

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。

把 $a_i - mid \times b_i$ 作为第 i 个物品的权值，贪心地选所有权值大于 0 的物品即可得到最大值。

为了方便初学者理解，这里放上完整代码：

参考代码

```
1  #include <algorithm>
2  #include <cmath>
3  #include <cstdio>
4  #include <cstdlib>
5  #include <cstring>
6  #include <iostream>
7  using namespace std;
8
9  int read() {
10     int X = 0, w = 1;
11     char c = getchar();
12     while (c < '0' || c > '9') {
13         if (c == '-') w = -1;
14         c = getchar();
15     }
16     while (c >= '0' && c <= '9') X = X * 10 + c - '0', c =
17     getchar();
18     return X * w;
19 }
20
21 constexpr int N = 100000 + 10;
22 constexpr double eps = 1e-6;
23
24 int n;
25 double a[N], b[N];
26
27 bool check(double mid) {
28     double s = 0;
29     for (int i = 1; i <= n; ++i)
30         if (a[i] - mid * b[i] > 0) // 如果权值大于 0
31             s += a[i] - mid * b[i]; // 选这个物品
32     return s > 0;
33 }
34
35 int main() {
36     // 输入
37     n = read();
38     for (int i = 1; i <= n; ++i) a[i] = read();
39     for (int i = 1; i <= n; ++i) b[i] = read();
40     // 二分
41     double L = 0, R = 1e9;
42     while (R - L > eps) {
43         double mid = (L + R) / 2;
44         if (check(mid)) // mid 可行, 答案比 mid 大
45             L = mid;
46         else // mid 不可行, 答案比 mid 小
47             R = mid;
48     }
49     // 输出
```

```

50     printf("%.6lf\n", L);
51     return 0;
    }

```

为了节省篇幅，下面的代码只保留 `check` 部分。主程序和本题是类似的。

POJ2976 Dropping tests

有 n 个物品，每个物品有两个权值 a 和 b 。

你可以选 $n - k$ 个物品 p_1, p_2, \dots, p_{n-k} ，使得 $\frac{\sum a_{p_i}}{\sum b_{p_i}}$ 最大。

输出答案乘 100 后四舍五入到整数的值。

把第 i 个物品的权值设为 $a_i - mid \times b_i$ ，然后选最大的 $n - k$ 个即可得到最大值。

```

1  bool cmp(double x, double y) { return x > y; }
2
3  bool check(double mid) {
4      int s = 0;
5      for (int i = 1; i <= n; ++i) c[i] = a[i] - mid * b[i];
6      sort(c + 1, c + n + 1, cmp);
7      for (int i = 1; i <= n - k; ++i) s += c[i];
8      return s > 0;
9  }

```

洛谷 4377 Talent Show

有 n 个物品，每个物品有两个权值 a 和 b 。

你需要确定一组 $w_i \in \{0, 1\}$ ，使得 $\frac{\sum w_i \times a_i}{\sum w_i \times b_i}$ 最大。

要求 $\sum w_i \times b_i \geq W$ 。

本题多了分母至少为 W 的限制，因此无法再使用上一题的贪心算法。

可以考虑 01 背包。把 b_i 作为第 i 个物品的重量， $a_i - mid \times b_i$ 作为第 i 个物品的价值，然后问题就转化为背包了。

那么 $dp[n][W]$ 就是最大值。

一个要注意的地方： $\sum w_i \times b_i$ 可能超过 W ，此时直接视为 W 即可。（想一想，为什么？）

```

1  double f[1010];
2
3  bool check(double mid) {

```

```

4   for (int i = 1; i <= W; i++) f[i] = -1e9;
5   for (int i = 1; i <= n; i++)
6       for (int j = W; j >= 0; j--) {
7           int k = min(W, j + b[i]);
8           f[k] = max(f[k], f[j] + a[i] - mid * b[i]);
9       }
10  return f[W] > 0;
11  }

```

POJ2728 Desert King

每条边有两个权值 a_i 和 b_i ，求一棵生成树 T 使得 $\frac{\sum_{e \in T} a_e}{\sum_{e \in T} b_e}$ 最小。

把 $a_i - mid \times b_i$ 作为每条边的权值，那么最小生成树就是最小值，

代码就是求最小生成树，故省略。

[HNOI2009] 最小圈

每条边的边权为 w ，求一个环 C 使得 $\frac{\sum_{e \in C} w}{|C|}$ 最小。

把 $a_i - mid$ 作为边权，那么权值最小的环就是最小值。

因为我们只需要判最小值是否小于 0，所以只需要判断图中是否存在负环即可。

另外本题存在一种复杂度 $O(nm)$ 的算法，如果有兴趣可以阅读 [这篇文章](#)。

```

1   int SPFA(int u, double mid) { // 判负环
2       vis[u] = 1;
3       for (int i = head[u]; i; i = e[i].nxt) {
4           int v = e[i].v;
5           double w = e[i].w - mid;
6           if (dis[u] + w < dis[v]) {
7               dis[v] = dis[u] + w;
8               if (vis[v] || SPFA(v, mid)) return 1;
9           }
10      }
11      vis[u] = 0;
12      return 0;
13  }
14
15  bool check(double mid) { // 如果有负环返回 true
16      for (int i = 1; i <= n; ++i) dis[i] = 0, vis[i] = 0;
17      for (int i = 1; i <= n; ++i)
18          if (SPFA(i, mid)) return true;
19      return false;
20  }

```

总结

分数规划问题是一类既套路又灵活的题目，一般使用二分解决。


分数规划问题的主要难点在于推出式子后想办法求出 $\sum w_i \times (a_i - mid \times b_i)$ 的最大值/最小值，而这个需要具体情况具体分析。

习题

- [JSOI2016 最佳团体](#)
- [SDOI2017 新生舞会](#)
- [UVa1389 Hard Life](#)
- [洛谷 P2868 \[USACO07DEC\] Sightseeing Cows G](#)

 本页面最近更新：2025/7/24 11:14:37，[更新历史](#)

 发现错误？想一起完善？ [在 GitHub 上编辑此页！](#)

 本页面贡献者：[StudyingFather](#), [Ir1d](#), [H-J-Granger](#), [countercurrent-time](#), [greyqz](#), [NachtgeistW](#), [Tiphereth-A](#), [Early0v0](#), [Enter-tainer](#), [Mout-sea](#), [AngelKitty](#), [banglee13](#), [CCXXI](#), [cjsoft](#), [diauweb](#), [ezoixx130](#), [GekkaSaori](#), [hsfzLZH1](#), [huaruoji](#), [Konano](#), [LovelyBuggies](#), [Makkiy](#), [mgt](#), [minghu6](#), [P-Y-Y](#), [PotassiumWings](#), [SamZhangQingChuan](#), [sshwy](#), [Suyun514](#), [weiyong1024](#), [alphagocc](#), [ChungZH](#), [GavinZhengOI](#), [Gesrua](#), [Henry-ZHR](#), [ksyx](#), [kxccc](#), [lyccrius](#), [lychees](#), [MicDZ](#), [ouuan](#), [Peanut-Tang](#), [r-value](#), [SukkaW](#), [Xeonacid](#)

© 本页面的全部内容在 [CC BY-SA 4.0](#) 和 [SATA](#) 协议之条款下提供，附加条款亦可能应用