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

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

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）：

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

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。

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

```cpp
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
using namespace std;

int read() {
  int X = 0, w = 1;
  char c = getchar();
  while (c < '0' || c > '9') {
    if (c == '-') w = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') X = X * 10 + c - '0', c = getchar();
  return X * w;
}

constexpr int N = 100000 + 10;
constexpr double eps = 1e-6;

int n;
double a[N], b[N];

bool check(double mid) {
  double s = 0;
  for (int i = 1; i <= n; ++i)
    if (a[i] - mid * b[i] > 0)  // 如果权值大于 0
      s += a[i] - mid * b[i];   // 选这个物品
  return s > 0;
}

int main() {
  // 输入
  n = read();
  for (int i = 1; i <= n; ++i) a[i] = read();
  for (int i = 1; i <= n; ++i) b[i] = read();
  // 二分
  double L = 0, R = 1e9;
  while (R - L > eps) {
    double mid = (L + R) / 2;
    if (check(mid))  // mid 可行，答案比 mid 大
      L = mid;
    else  // mid 不可行，答案比 mid 小
      R = mid;
  }
  // 输出
```

```
  printf("%.6lf\n", L);
  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 $p_1, p_2, \cdots, p_{n-k}$，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。

```
bool cmp(double x, double y) { return x > y; }

bool check(double mid) {
  int s = 0;
  for (int i = 1; i <= n; ++i) c[i] = a[i] - mid * b[i];
  sort(c + 1, c + n + 1, cmp);
  for (int i = 1; i <= n - k; ++i) s += c[i];
  return s > 0;
}
```

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？）

```
double f[1010];

bool check(double mid) {
```

```
  for (int i = 1; i <= W; i++) f[i] = -1e9;
  for (int i = 1; i <= n; i++)
    for (int j = W; j >= 0; j--) {
      int k = min(W, j + b[i]);
      f[k] = max(f[k], f[j] + a[i] - mid * b[i]);
    }
  return f[W] > 0;
}
```

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。

```
int SPFA(int u, double mid) {  // 判负环
  vis[u] = 1;
  for (int i = head[u]; i; i = e[i].nxt) {
    int v = e[i].v;
    double w = e[i].w - mid;
    if (dis[u] + w < dis[v]) {
      dis[v] = dis[u] + w;
      if (vis[v] || SPFA(v, mid)) return 1;
    }
  }
  vis[u] = 0;
  return 0;
}

bool check(double mid) {  // 如果有负环返回 true
  for (int i = 1; i <= n; ++i) dis[i] = 0, vis[i] = 0;
  for (int i = 1; i <= n; ++i)
    if (SPFA(i, mid)) return true;
  return false;
}
```

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
