基本技巧——分数规划 学习笔记

引入

分数规划用来求一个分式的极值。

具体的，给定 n 个元素，每个元素有属性 a_i, b_i，求一个集合 $P \in [1, n]$，最大／最小化比率：

$$\frac{\sum_{i \in P} a_i}{\sum_{i \in P} b_i}$$

求解

二分法

假设我们要求最大值（求最小值的方法和求最大值的方法类似），二分一个 mid，然后推式子（省略范围）：

$$\begin{aligned} & \sum a_i / \sum b_i \geq \mathrm{mid} \\ \Longrightarrow \quad & \sum a_i \geq \sum b_i \times \mathrm{mid} \\ \Longrightarrow \quad & \sum a_i - \sum b_i \times \mathrm{mid} \geq 0 \\ \Longrightarrow \quad & \sum (a_i - b_i \times \mathrm{mid}) \geq 0 \end{aligned}$$

设 $g(x) = \sum (a_i - b_i \times x)$，则原问题其实就是最大化 $g(x)$。可以用二分求解：二分一个 mid，如果 $g(\mathrm{mid}) \geq 0$，则说明可行；否则就是不可行。

实际求解的时候一般是，如果 $g(\mathrm{mid}) > 0$，那么就 $l \leftarrow \mathrm{mid} + 1$，否则 $r \leftarrow \mathrm{mid}$。

在题目中，一般来说，分数规划的难点在于求解 $g(x)$ 的最大／最小值。

模型

01 分数规划

题目：`POJ2976 Dropping tests`

题意：在 n 个物品中选取 k 个，使得比率最大。

分析：二分最大的比率为 mid，最大化 $g(\mathrm{mid})=\sum(a_i-b_i\times\mathrm{mid})$；可以贪心来考虑，将 $a_i-b_i\times\mathrm{mid}$ 视为物品的权值，将权值前 k 大的数加起来，即 $g(\mathrm{mid})$；如果 $g(\mathrm{mid})>0$，则 $l\leftarrow\mathrm{mid}+1$，否则 $r\leftarrow\mathrm{mid}$。

变式：选取若干个，最大化比率。贪心考虑，只取所有权值为正的物品，其余同上。

代码：

```
using db = double;
const db eps = 1e-6;
int n, k, a[1010], b[1010];
db c[1010];
inline bool cmp(const db a, const db b) {
    return a > b;
} bool check(const db x) {
    for (int i = 1; i <= n; ++i) c[i] = a[i] - b[i] * x;
    sort(c + 1, c + n + 1, cmp); db s = 0;
    for (int i = 1; i <= n - k; ++i) s += c[i];
    return s > 0;
} signed main() {
    while (1) {
        n = rr, k = rr; if (!n && !k) break;
        for (int i = 1; i <= n; ++i) a[i] = rr;
        for (int i = 1; i <= n; ++i) b[i] = rr;
        db l = 0, r = 100, mid; while (r - l > eps) {
            mid = (l + r) / 2;
            if (check(mid)) l = mid;
            else r = mid;
        } printf("%d\n", int(mid * 100 + 0.5));
    } return 0;
}
```

最优比率生成树

题目：POJ2728 Desert King

题意：给定一棵图，每条边有两个权值 a_i,b_i，求一棵生成树，使得 $\dfrac{\sum_{e\in T}a_e}{\sum_{e\in T}b_e}$ 最小。

分析：与上一题很类似，把 $a_i-b_i\times\mathrm{mid}$ 作为每条边的权值，那么这个图的最小生成树就是 $f(\mathrm{mid})$ 最小值

代码：略。

最优比率生成环

题目：P3199 最小圈、P1768 天路

题意：给定一个图，每条边有价值 v_i 和费用 p_i，求一个环，使得总价值与总费用的比值最小。

分析：我们二分一个答案 mid，如果存在一个环，满足 $\sum v/\sum p \ge \text{mid}$，那么移项一下就有 $\sum v \ge \text{mid} \times \sum p$，就有 $\sum v \ge \sum(\text{mid} \times p)$，也即存在一个环满足 $\sum(\text{mid} \times p - v) \le 0$，也就是把 $\text{mid} \times p - v$ 作为新的边权，要判断图中是否存在负环。找负环可以用 SPFA 的 DFS 版本。

代码：

```cpp
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 1;
    return 0;
}
```

应用

例题

题目：AT_abc324_f Beautiful Path（考前打比赛遇到的）。

题意：给定一个 n 个点 m 条边的有向图，保证 $\forall i \in [1,n] : u_i < v_i$。每条边有价值 b_i 和费用 c_i，找一条路径 $1 \to n$，最大化 $\sum b_i/\sum c_i$。

分析：二分一个答案 ans，则有 $\sum b_i/\sum c_i \ge \text{ans} \longrightarrow \sum b_i \ge \text{ans} \times \sum c_i \longrightarrow \sum b_i - \text{ans} \times \sum c_i \ge 0 \longrightarrow \sum(b_i - \text{ans} \times c_i) \ge 0$，因此，以 $b_i - \text{ans} \times c_i$ 为边权建图跑最短路即可。

代码:

```
const int N = 2e5 + 10;
const db eps = 1e-10, db INF = 1e18;
int n, m; struct node {
    int v, b, c;
}; vector<node> g[N];
db f[N]; bool check(db x) {
    for (int i = 1; i <= n; ++i) f[i] = -INF;
    f[1] = 0; for (int i = 1; i <= n; ++i)
        for (node j : g[i]) f[j.v] = max(f[j.v], f[i] + j.b - j.c * x);
    return f[n] > 0;
} signed main() {
    n = rr, m = rr;
    for (int i = 1, u, v, b, c; i <= m; ++i) {
        u = rr, v = rr, b = rr, c = rr;
        g[u].push_back({v, b, c});
    } db l = 0, r = 1e4; while (r - l > eps) {
        db mid = (l + r) / 2;
        if (check(mid)) l = mid;
        else r = mid;
    } printf("%.15lf", l);
    return 0;
}
```

练习题

见：https://www.luogu.com.cn/training/400462

Reference

[1] https://oi-wiki.org/misc/frac-programming/

[2] https://www.cnblogs.com/captain1/p/9929128.html
