分数规划

1 分数规划

先给出分数规划一般形式：

$$\textbf{Minimize} \quad \lambda = f(\mathbf{x}) = \tfrac{a(\mathbf{x})}{b(\mathbf{x})}(\mathbf{x} \in S)$$
$$s.t. \ \forall \mathbf{x} \in S, b(\mathbf{x}) > 0$$

其中，解向量 **x** 在解空间 S 中，$a(\mathbf{x})$ 与 $b(\mathbf{x})$ 都是**连续的实数**。

解决分数规划问题的一般方法是分析其对偶问题，但更加实用的方法是对其进行**参数搜索**，即对答案进行猜测，然后验证猜测值的最优性，将问题转化为判定性问题或其他的优化问题。由于分数规划模型的特殊性，使得能够构造另外一个由猜测值 λ 作为自变量的相关问题，且该问题的解一般满足一定**单调**性，或其他的可以减少参数搜索范围的性质，从而逼近答案。

比如 **Dinkelbach** 算法，每次直接把上次子问题的解向量代入原问题的表达式，算出下一个迭代式的猜测值。

假设 $\lambda_0 = f(\mathbf{x_0})$ 是最优解，根据定义有

$$\lambda_0 = f(\mathbf{x_0}) = \tfrac{a(\mathbf{x_0})}{b(\mathbf{x_0})}$$
$$\Rightarrow \lambda_0 \cdot b(\mathbf{x_0}) = a(\mathbf{x_0})$$
$$\Rightarrow a(\mathbf{x_0}) - \lambda_0 \cdot b(\mathbf{x_0}) = 0$$

由上面的形式构造一个新函数

$$g(\lambda) = \min_{\mathbf{x} \in S}[a(\mathbf{x}) - \lambda \cdot b(\mathbf{x})]$$

这个新函数是一个非分式的规划。先来挖掘函数 $g(\lambda)$ 本身的性质。

性质 1.1（单调性） *$g(\lambda)$ 是一个严格递减函数，即，* $\forall \lambda_1, \lambda_2 \in \mathbb{R}, \lambda_1 < \lambda_2 \ \Leftrightarrow \ g(\lambda_1) > g(\lambda_2)$

有了单调性，就意味着可以采取二分搜索的办法逼近答案。但还不知道目标是什么，下面考察构造出的新函数与原目标函数的最优解关系。

定理 1.2(Dinkelbach 定理) *若 λ_0 是原问题的最优解，则 $g(\lambda) = 0$ 当且仅当 $\lambda = \lambda_0$*

证明：

（1）先证**必要性** $\lambda = \lambda_0 \ \Rightarrow \ g(\lambda) = 0$：设 $\lambda_0 = f(\mathbf{x_0})$ 为原规划的最优解，则 $g(\lambda_0) = 0$。

对于 $\forall \mathbf{x} \in S$，都不会比 $\mathbf{x_0}$ 优：

$$\lambda_0 = \frac{a(\mathbf{x_0})}{b(\mathbf{x_0})} \leq \frac{a(\mathbf{x})}{b(\mathbf{x})} \Rightarrow a(\mathbf{x}) - \lambda_0 \cdot b(\mathbf{x}) \geq 0$$

然而 $\mathbf{x_0}$ 可以取到这个下限。

$$\lambda_0 = \frac{a(\mathbf{x_0})}{b(\mathbf{x_0})} \Rightarrow a(\mathbf{x_0}) - \lambda_0 \cdot b(\mathbf{x_0}) = 0$$

故 $g(\lambda_0)$ 的最小值由 $\mathbf{x_0}$ 确定，$g(\lambda_0) = 0$。

(2) 再证**充分性** $g(\lambda) = 0 \ \Rightarrow \ \lambda = \lambda_0$

反证法。反设存在一个解 $\lambda' = f(\mathbf{x}')$ 是更优的解，根据定义有

$$\lambda' = \tfrac{a(\mathbf{x}')}{b(\mathbf{x}')} < \lambda$$
$$\Rightarrow a(\mathbf{x}') - \lambda \cdot b(\mathbf{x}') < 0$$

那么，将 $\mathbf{x}'$ 代入，可以发现 $g(\lambda)$ 一定是小于零的，与题设矛盾。

由性质 1.1 与定理 1.2, 容易得到下面的推论。

推论 1.3 设 λ_0 为该规划的最优解，则

$$\begin{cases} g(\lambda) = 0 \Leftrightarrow \lambda = \lambda_0 \\ g(\lambda) < 0 \Leftrightarrow \lambda > \lambda_0 \\ g(\lambda) > 0 \Leftrightarrow \lambda < \lambda_0 \end{cases}$$

由推论 1.3，就可以对最优解 λ_0 进行二分查找逼近。每次查找一个猜测值，计算 $g(\lambda)$ 的值，而 $g(\lambda)$ 是一个非分数规划，将原问题简单化了，便可以设计出其他有效的算法解决这个规划。

以上分析是针对最小化目标函数的分数规划，对于最大化目标函数也是同样道理。

2 0-1 分数规划

分数规划的一个特例是 **0-1 分数规划**，就是其解向量 $\mathbf{x}$ 满足 $\forall x_i \in \{0, 1\}$，形式化如下：

$$\min_{\mathbf{x} \in S} f(\mathbf{x}) = \frac{\sum_i a_i x_i}{\sum_i b_i x_i} = \frac{a \cdot \mathbf{x}}{b \cdot \mathbf{x}}$$

同样地，必须要满足 $\forall \mathbf{x} \in S, \mathbf{b} \cdot \mathbf{x} > 0$，且有比较特殊的 $S \subseteq \{0, 1\}^n$

3 0-1 分数规划应用

0-1 分数规划问题主要包含一般的 **0-1 分数规划**、**最大密度子图**、**最优比率生成树问题**、**最优比率环问题**等。我们将会对这四个问题进行讨论。

3.1 0-1 分数规划

题目：*A2039 0-1 分数规划*

【**题目大意**】：给出 n 对 (ai,bi)，选取其中 k 对，求 ∑ai/∑bi 的最大值，最小值。

【**解析**】

设 $Ans = \frac{\sum a[i]*x[i])}{\sum b[i]*x[i]}$，且 Ans 为最大值。也就是说对于说对于任意组合满足 $\sum a[i] * x[i] - \lambda \sum b[i] * x[i] \leq 0$, 即求 $a[i] - \lambda * b[i]$ 的最大 k 组。若大于大于 0，则增大 λ；若小于 0，则减小 λ。

二分答案参考代码:

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N=1e4+10; const double eps=1e-5,INF=1e9;
int n,k; double a[N],b[N],c[N];
#define in read()
inline int read(){
  int f=1,k=0;char cp=getchar();
  while(cp!='-'&&(cp<'0'||cp>'9')) cp=getchar();
  if(cp=='-') f=-1,cp=getchar();
  while(cp>='0'&&cp<='9') k=(k<<3)+(k<<1)+cp-48,cp=getchar();
  return f*k;
}
inline bool check1(const double mid){
  for(int i=1;i<=n;i++) c[i]=a[i]-mid*b[i];
  sort(c+1,c+n+1);
  double res=0; for(int i=1;i<=k;i++) res+=c[i];
  return res<0;
}
inline bool check2(const double mid){
  for(int i=1;i<=n;i++) c[i]=a[i]-mid*b[i];
  sort(c+1,c+n+1);
  double res=0; for(int i=1;i<=k;i++) res+=c[n-i+1];
  return res>0;
}
int main(){
  n=in,k=in;
  for(int i=1;i<=n;i++) a[i]=in;
  for(int i=1;i<=n;i++) b[i]=in;
  double l=0,r=1;
  while(l+eps<r){ double mid=(l+r)/2.0;
    if(check1(mid)) r=mid;
    else l=mid;
  }
  printf("%.4lf\n",l);
  l=0,r=1;
```

```
    while(l+eps<r){
      double mid=(l+r)/2.0;
      if(check2(mid)) l=mid;
      else r=mid;
    }
    printf("%.4lf\n",l);
    return 0;
}
```

迭代法：（Dinkelbach 算法）

随意构造一个值 x, 令 d[i]=a[i]-x*b[i]，将 d[i] 排序，然后选出排序后的前 k 组，计算新的 $x' = \frac{\sum a[i]}{\sum b[i]}$, 如果 x, x' 在精度范围内则 x 即是答案，否则令 $x = x'$ 继续迭代。

参考代码：

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e4+100;
const double eps=1e-5;
struct node{
    int a, b;
    double c,d;
}a[N];
int n,k;
double ans1,ans2;
int cmp1(const node& x,const node& y){return x.c < y.c;}
int cmp2(const node& x,const node& y){return x.d > y.d;}
int main(){
    cin>>n>>k;
    for(int i=1;i<=n;i++)cin>>a[i].a;
    for(int i=1;i<=n;i++)cin>>a[i].b;
    double x1=0.5,x2=0.5;
    int ok1=0,ok2=0;
    while(1){
        if(ok1&&ok2)break;
        if(ok1==0){//最小值计算
            ans1=x1;
            for(int i=1;i<=n;i++)a[i].c=(double)a[i].a - x1*a[i].b;
            sort(a+1,a+n+1,cmp1);
```

```
            double suma=0.0,sumb=0.0;
            for(int i=1;i<=k;i++)suma+=a[i].a,sumb+=a[i].b;
            x1 = suma/sumb;
            if(fabs(ans2-x2)<eps)ok1=1;
        }
        if(ok2==1)continue;

        ans2=x2;
        for(int i=1;i<=n;i++)a[i].d=(double)a[i].a - x2*a[i].b;
        sort(a+1,a+n+1,cmp2);
        double suma=0.0,sumb=0.0;
        for(int i=1;i<=k;i++)suma+=a[i].a,sumb+=a[i].b;
        x2 = suma/sumb;
        if(fabs(ans2-x2)<eps)ok2=1;

    }
    printf("%.4lf\n%0.4lf",ans1,ans2);
    return 0;
}
```

3.2 最优比率生成树

题目：*A2066.*「模板」「最优比率生成树」重修家园

题意：N 个点 m 条边无向图，边有 2 个权值 ci,di 求一个生成树，要求 r=∑ci/∑di 最小（大）

解析：

我们所求的比率 $r = \frac{\sum c[i]*x[i]}{\sum d[i]*x[i]}, 1 \le i \le m$。设 x[i] 等于 1 或 0，表示边 e[i] 是否属于生成树。为了使 r 最大，设计一个子问题 → 设 z(L)=∑(cost[i]*x[i])-L*∑(dis[i]*x[i]) 最大，其中 d[i]=cost[i]-L*dis[i])。我们可以把 z(L) 看做以 d 为边权的最小生成树的总权值。

理解：求最小

设最优为 Ans，则对于任意生成树都有

(∑cost[i]*x[i])/(∑dis[i]*x[i])>=Ans，即 ∑(cost[i]*x[i])-Ans*∑(dis[i]*x[i])>=0，故应该求最小生成树。

同理如果是求最大就是最大生成树

具体算法：二分 + 最小生成树，二分比率值 mid，然后建图，在原图基础上，每条边权为：-ci-mid*ti，然后求最大生成树。最后判断最大生成树边权和 +F 是否 >=0

```
#include<bits/stdc++.h>
using namespace std;
```

```
#define ll long long
const double eps=1e-8;
const int N=1e5+5;
const int M=1e6+20;
int u[M],v[M],c[M],d[M],n,m,F;
struct edge{
  int u,v;double w;
}e[M];
int fa[N];
int cmp(edge a,edge b){
  return a.w>b.w;
}
int find(int x){
  return x==fa[x]?x:fa[x]=find(fa[x]);
}
int check(double mid){
  for(int i=1;i<=n;i++)fa[i]=i;
  for(int i=1;i<=m;i++)e[i].u=u[i],e[i].v=v[i] ,e[i].w=-(double)c[i
   ]-mid*d[i];
  int cnt=0;
  double ret=0;
  sort(e+1,e+m+1,cmp);
  for(int i=1;i<=m;i++){
    int fx=find(e[i].u),fy=find(e[i].v);
    if(fx==fy)continue;
    fa[fx]=fy;ret+=e[i].w;
    cnt++;if(cnt==n-1)break;
  }
  return F+ret>=0;
}
int main(){
  scanf("%d%d%d",&n,&m,&F);
  for(int i=1;i<=m;i++)
    scanf("%d%d%d%d",u+i,v+i,c+i,d+i);

  double  l=0.0,r=100000;
  while(r-l>eps){
```

```
    double mid=(l+r)/2.0;
    if(check(mid))l=mid;else r=mid;
  }
  printf("%.4lf",l);
  return 0;
}
```

3.3 最优比率生成环

题目：*P3076, 观光奶牛 Sightseeing Cows*

题意：求一个环，使得点权和除以边权和最大。

解析：

令在一个环里，点权为 v[i]，对应的边权为 e[i]，即要求：∑v[i]/∑e[i] 最大的环。设题目答案为 ans，即对于所有的环都有 ∑v[i]/∑e[i]<=ans；

变形得：ans*∑e[i]>=∑v[i]；

再得：∑(ans*e[i]-v[i])>=0；

稍分析一下，就有：

当 k<ans 时，就存在至少一个环 ∑(k*e[i]-v[i])<0，即有负权回路；

当 k>=ans 时，就对于所有的环 ∑(k*e[i]-v[i])>=0，即没有负权回路。

算法：重新建图，使得边权为 k*e[i]-v[i]。用 SPFA 算法，二分枚举 k，判断是否存在负权回路，若存在，说明 k 偏小了，则增大 k，若不存在，则减小 k。

注意：对于 $v[i]$ 和 $k*e[i]$ 在符号中的先后关系需要看题目是求最大值还是最小值，此题是把 $k*e[i]$ 放在符号前面的。

3.4 最大密度子图

题目：*bzoj1312. Hard Life 生活的艰辛*

描述：ADN 公司内部共 n 个员工，员工之间可能曾经因为小事有了过节，总是闹矛盾。若员工 u 和员工 v 有矛盾，用边 (u, v) 表示，共 m 个矛盾。最近，ADN 公司内部越来越不团结，Amber 决定裁员。Amber 想得到一个被裁人员的清单，使得被裁人员间的不团结率最高。不团结率定义为被裁人员间的矛盾总数与被裁人员数的比值（不团结率 = 被裁人员之间的矛盾总数 / 被裁人员数）