0 1分数规划_良弓藏_新浪博客

http://wenku.baidu.com/view/f2a563d076eeaeaad1f3305e.html
http://hi.baidu.com/misshanli/blog/item/adddc2ffe6d959314e4aea6a.html

01分数规划：
给出n个分数ai/bi，选出m个（m<=n）
使得∑（ai）/∑（bi）{此中i是被选出的分数编号}达到最大值。

01 分数规划的两种办法：
1、二分法
二分一个答案k，令ci=ai-k*bi，并将ci排序，
选出最大的m个，若是∑（ci）{1<=i<=m}>=0，那么进步答案k的下界，
不然降落上界，直到k的精度满足请求为止。
∑（ai）/∑（bi）>= x
=> ∑（ai）>= ∑（x*bi）
=> ∑（ai）- ∑（x*bi）>= 0
=> ∑（ai - x*bi）>= 0

2、Dinkelbach迭代法
随便机关一个答案k，令ci=ai-k*bi，并将ci排序，选出最大的m个，
令q=∑（ai）/∑（bi）{1<=i<=m}与k的差在精度局限内就输出，
不然令k=q，直至满足精度请求。
总结：两种办法各有利弊，二分法精度较高，易懂，速度较慢。
Dinkelbach迭代法速度较快，但精度低，不易懂得和证实。

```
// 二分法
#include
#include
#include
#include
using namespace std;

#define N 1100
#define EPS 1e-6
#define EQ(a, b) (fabs(a-b)<=EPS)
#define FFOR(i, n) for(i=0; i

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

bool compare(double a, double b){
return a>b;
}

int ques(double x){
int i;
FFOR(i, n) c[i]=a[i]-x*b[i];
sort(c, c+n, compare);
double res=0;
FFOR(i, n-k) res+=c[i];
return (res>=0 ? 1:0);
}

int solve(){
```

```
double up, down, mid, ok;
up=1.0; down=0.0;
while(!EQ(up, down)){
mid=(up+down)/2;
ok=ques(mid);
if(ok) down=mid;
else up=mid;
}
return (int)(up*100+0.5);
}

int main(){
int i;
while(scanf("%d%d", &n, &k)&& n|k){
FFOR(i, n) scanf("%lf", &a[i]);
FFOR(i, n) scanf("%lf", &b[i]);
printf("%d\n", solve());
}
return 0;
}

// 迭代法
#include
#include
#include
#include
using namespace std;

#define N 1100
#define EPS 1e-6
#define EQ(a, b) (fabs(a-b)<=EPS)
#define FFOR(i, n) for(i=0; i

struct info{ double v; int i; }c[N];

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

bool compare(info a, info b){
return a.v>b.v;
}

double ques(double x){
int i;
FFOR(i, n) {c[i].v=a[i]-x*b[i]; c[i].i=i; };
sort(c, c+n, compare);

double fz, fm;
fz=fm=0;
FFOR(i, n-k) { fz+=a[c[i].i]; fm+=b[c[i].i];}
return fz/fm;
}

int solve(){
double k, q;

k=0.0, q=rand(); // q=0.5 效果会好一点
```

```
while(!EQ(k, q)){
k=q;
q=ques(k);
}
return (int)(k*100+0.5);
}

int main(){
int i;
while(scanf("%d%d", &n, &k)&& n|k){
FFOR(i, n) scanf("%lf", &a[i]);
FFOR(i, n) scanf("%lf", &b[i]);
printf("%d\n", solve());
}
return 0;
}
```