Hello World ! ---二分三分

/\*

Pi 最好用acos(-1.0)

ppt里教了一个好方法，先扩大半径100000再来搞，减少精度损失，mark.

\*/

/\*

相似三角形

为什么会出现在二分三分专题里？？

记一下相似三角形的性质：

对应角相等

对应边，对应高成比例（称为相似比）

周长成比例（等于相似比）

面积成比例（相似比的平方）

/\*

二分答案 简单题 好WA点

输出方案WA了，没考虑到“一定要分成K组”!!! 好WA点

\*/

/\*

二分三分 卡精度

题意：一个由一个矩形+两个相同弓形组成的操场，且这两个弓形是属于同一个圆。已知周长为400，

矩形长宽比，求矩形的长宽。

思路：既然两个弓形属于同一个圆，那么该圆心就是矩形的对角线交点

二分枚举矩形的长边l ，判断与400的大小关系就可以了。

说实话，我列出了式子，但不知道怎么判断它是凸函数还是单调函数=\_= 怎么办=\_=

卡精度，我是用周长 \* 10000来搞的，再加个dblcmp()

弧的周长l = R \* θ

l = n（圆心角）x π（圆周率）x r（半径）/180

l =α(圆心角弧度数)× r（半径）

\*/

#define Pi ( acos(-1.0) )

int a, b;

const double perimeter = 400 \* 10000;

double calc(double l)

{

double h = b\*l/a, r = sqrt(l\*l+h\*h)/2;

return l + r\*2\*atan(h/l) - perimeter/2;

}

/\*

补三角知识：

sin(Pi/n) = r/(R-r)

-> r = R\*sin(Pi/n) / (1+sin(Pi/n))

画个图，R-r就是大圆心到小圆心的距离，做小圆切线l交于大圆心，再做小圆心到l的垂线，就会发现

sin(Pi/n) = r/(R-r)

\*/

#define Pi ( acos(-1.0) )

double r = R \* sin(Pi/n) / (1 + sin(Pi/n));

/\*

二分三分 二分统计 二分搜索 好题 经典题 搜索剪枝

题意：给n(1<=n<=30)个数wi>=0, 取出若干个，要求和不超过W。问合法方案数。

思路：折半统计

把n个数分成两份，分别二进制枚举合法的情况(W >= sum)，分别存到两个堆里，其中一个堆

sort一遍。然后我们就可以枚举一个堆的状态ai，再在另一个堆里二分找出所有ai+bi<= W的

bi。 其实就是把a堆和b堆合并起来。

折半法把复杂度2^30直接降到2^15，太爽了，拒Neko说这是个套路题=\_= 跟前两天我们讨论

那道“5个堆各取一个数，使之和为0”的题hdu\_4334很像。

练习赛的时候neko和fubao用暴力搜索+剪枝过了...可我们拉的是二分三分专场呀=\_=

剪枝有这么两个：

(1)sort，应该来说把大的放前面比较能减少冗余递归；

(2)dfs时发现后缀和加上当前sum都不超过W，则用组合数学算一下，就回溯，不前进了；

(3)当前sum == W，就退出。但是这个不能用在从大到小排序上，因为wi可以为0. 所以这个

剪枝慎用

\*/

int solve()

{

int half = n/2;

na = nb = 0;

for(int i = 0, oo = (1 << half); i < oo; i++) { //前部分n/2

ll sum = 0;

for(int j = 0; j < half; j++) if((1 << j) & i) sum += w[j];

if(sum <= W) a[na++] = (int)sum;

}

for(int i = 0, oo = (1 << (n-half)); i < oo; i++) {//后部分n-n/2

ll sum = 0;

for(int j = 0; j < n-half; j++) if((1 << j) & i) sum += w[j+half];

if(sum <= W) b[nb++] = (int)sum;

}

sort(b, b+nb);

int ans = 0;

for(int i = 0; i < na; i++)

ans += upper\_bound( b, b+nb, W-a[i] ) - b; //合并 用二分

return ans;

}

/\*

算N!有多少个0

算N的阶层有多少个0..............啊啊啊啊么心情写题解

直接摘：就看这n个数中有多少个5的倍数，因为1-n中肯定有足够多的偶数能将5的倍数转换为10的倍

数，所以有多少个5的倍数阶层就有多少个0。

循环：因为如果像25\*4，50\*2,都能产生2个0，所以我们要找出25的倍数，因为前面已经找出了除以

了5，后面在除以5就是25的倍数了，并且因为前面已经25,50已经因为是5的倍数算过一次，后面在

算的时候直接加25的倍数的个数就ok了

\*/

int zeros(ll x)

{

return x == 0 ? 0 : x/5 + zeros(x/5);

}

int main()

{

while(cases--) {

scanf("%d", &Q);

ll l = 1, r = (1LL << 30), mid, ans = -1;

while(l <= r) {

mid = (l + r) / 2;

int res = zeros(mid) ;

if(res == Q) ans = mid, r = mid - 1;

else if(res < Q) l = mid + 1;

else r = mid - 1;

}

if(ans == -1) printf("Case %d: impossible\n", ++Cas);

else printf("Case %d: %lld\n", ++Cas, ans);

}

}

/\*

卡特兰数　+ 乘法逆元 + 二分三分 扩展欧几里得 好题

题意：给a, b (1 <= a <= b <= 10^10)，设a，b之间有n个完全数[x>1,y>1，使得m=x^y，则m为完全

数，用这n个数作为结点，求这n个结点能形成多少种二叉树？

思路：第一次用乘法逆元 ~~ 乘法逆元有可能为负数，要记得 (inv % MOD + MOD) % MOD !!!

catalan number : 1, 1, 2, 5, ... 不过这里n=0, ans=0

+

摘：

乘法逆元：

x\*y ≡ 1mod (mod)，则称 x 是 y 对于mod的乘法逆元

分数取模就要用到了，因为上面递推式有分母i+1

要求(i+1)^-1 % mod = ?

就等价于i+1的逆元x%mod了

令y = i+1，x\*y ≡ 1mod (mod) → x\*y + k\*mod == 1

用扩展欧几里德即可算出y的逆元x

扩展欧几里得：http://972169909-qq-com.iteye.com/blog/1140914

最后二分查找a，b之间有多少个完全数

\*/

//h(n)=h(n-1)\*(4\*n-2)/(n+1);

void init()

{

cat[0] = 0, cat[1] = 1;

for(int i = 2; i < 1000005; i++) {

cat[i] = (cat[i-1]\*(4\*i-2) % MOD \* ((mod\_inv(i+1)%MOD+MOD)%MOD)) % MOD; //!!!

}

n = 0;

for(ll i = 2; i < 100000; i++) {

ll j = i\*i;

while(j <= 10000000000LL) {

p[n++] = j;

j \*= i;

}

}

sort(p, p+n);

int idx = n; n = 1;

for(int i = 1; i < idx; i++) if(p[i] != p[i-1]) p[n++] = p[i];

}

/\*

二分三分 预处理 hash 判重 好题

考点是预处理...neko和fuyuchang用搜索50ms，我...500+ms =\_= 还以为我用队列轮换搞会比较快，

但是map判重巨慢=\_= 换了Leonardo B的hash，快了300ms !!! 果断收了hash table~

题意：lucky number是由4和7组成的数，very lucky number是由多个lucky number相乘的积，包括

lucky number在内，问A~B内有多少个very lucky number

思路：难点在预处理出所有very lucky number

fuyuchang和neko搜索不会，Leonardo B的的每一步都unique一遍来去重不优美，我写了个用

队列轮换搞+map/hash去重

方法是这样的，(1)先暴利预处理出lucky number. 不要用deque，会爆，还没到1000就爆了

(2)依旧暴利预处理出very lucky number ，只不过要判重

\*/

typedef unsigned int u32;

typedef unsigned char uch;

#define MAX\_QUEUE 346375

#define Limit 1000000000000LL

map<ll, bool> M;

ll lucky[MAX\_QUEUE], Q[MAX\_QUEUE];

int lucky\_head, Q\_head;

#define HASHB 100007//2048

struct HM {

typedef ll Datum; typedef vector<Datum> DV; DV b[HASHB];

u32 fnv\_hash(Datum \*k, int len) const {

uch \*p = reinterpret\_cast<uch\*>(k);

u32 h = 2166136261U;

for (int i = 0; i < len; ++i) h = (h \* 16777619U ) ^ p[i];

return h;

}

void add(Datum k) {

int i = fnv\_hash(&k, sizeof(Datum)) % HASHB;

for (int j = 0, J = b[i].size(); j < J; ++j)

if (b[i][j] == k) return;

b[i].push\_back(k);

}

bool get(Datum k) {

int i = fnv\_hash(&k, sizeof(Datum)) % HASHB;

for (int j = 0, J = b[i].size(); j < J; ++j)

if (b[i][j] == k) return true;

return false;

}

} hm;

void init()

{

int head = 0, tail = 0;

int times = 12;

lucky[head++] = 0;

while(times--) {

int tmp\_head = head;

for(; tail < tmp\_head; tail++) {

if(lucky[tail] \* 10 + 4 <= Limit) lucky[head++] = lucky[tail]\*10 + 4;

if(lucky[tail] \* 10 + 7 <= Limit) lucky[head++] = lucky[tail]\*10 + 7;

}

}

lucky\_head = head;

head = 0, tail = 0;

Q[head++] = 1;

while(true) {

int tmp\_head = head;

if(tail == head) break;

for( ; tail < tmp\_head; tail++) {

for(int i = 1; i < lucky\_head; i++) {

ll x = lucky[i] \* Q[tail];

if(x <= Limit) {

//if(!M[x]) M[x] = true, Q[head++] = x;

if(!hm.get(x)) hm.add(x), Q[head++] = x;

}

else break;

}

}

}

Q[0] = Limit + 4;

sort(Q, Q+head);

Q\_head = head - 1;

}

/\*

01分数规划 二分三分

orz ... neko那混蛋，跟他想了几天，没搞头，然后刚才他突然说，“这不就是01规划嘛”

然后...然后真想揍他...

http://m.blog.csdn.net/blog/kksleric/7526239

\*/

double calc(double x)

{

for(int i = 0; i < n; i++) a[i].k = a[i].p - a[i].m\*x;

sort(a, a+n);

double sum = 0;

for(int i = 0; i < d; i++) sum += a[i].k;

return sum;

}

int main()

{

while(cases--) {

scanf("%d%d", &n, &d); d = n - d;

double l = 0, r = 1, mid;

for(int i = 0; i < n; i++) scanf("%lf%lf", &a[i].p, &a[i].m), r += a[i].p;

while(r - l > EPS) {

mid = (r + l) / 2;

if(calc(mid) > 0) l = mid;

else r = mid;

}

printf("Case %d: %.8f\n",++Cas, mid\*100+EPS);

}

return 0;

}

/\*

二分三分 二分答案 两次二分 经典题 好题

题意：Given a N × N matrix A, whose element in the i-th row and j-th column Aij is an

number that equals i^2 + 100000 × i + j^2 - 100000 × j + i × j, you are to find the

M-th smallest element in the matrix.

思路：竟然是二分答案，太神奇了

容易看出来{Aij}按行递增，列是个凸函数(二次函数)

二分答案，再求每一列上有多少个数小于等于该解，而根据列递增我们又可以来一次二分

两次二分，所以复杂度O(logn \* n \* logn)

算每一列小于等于该解的办法还可以用一元二次不等式来搞。

扩展：现在给的式子是f(i, C)递增，f(C, j)凸函数，那要是给的式子f(i, C)也是凸函数呢？

还是可以用二分答案 + 枚举列，但算每一列小于该解的数目就只能用解二次不等式搞了，不能

再二分了。

3204ms，挺慢，long long计算太多，可以常数级优化一下

\*/

long long calc(long long col, long long x)

{

long long tmp1 = (100000+col), tmp2 = (col\*col-100000\*col);

long long r = N, l = 1, mid, ans = 0;

while(l <= r) {

mid = (l + r) / 2;

if( mid\*mid + tmp1\*mid + tmp2 <= x )

ans = mid, l = mid + 1;

else r = mid - 1;

}

return ans;

}

bool check(long long x)

{

long long sum = 0;

for(int col = 1; col <= N; col++) sum += calc((long long)col, x);

return sum >= M;

}

int main()

{

while(cases--) {

scanf("%I64d%I64d", &N, &M);

long long r = (1LL << 62), l = -(1LL << 62), ans, mid;

while(l <= r) {

mid = (l + r) / 2;

if(check(mid)) ans = mid, r = mid - 1;

else l = mid + 1;

}

printf("%I64d\n", ans);

}

return 0;

}

/\*

二分三分

摘:

本题要求影子最长的长度。

在灯、人头、墙角成一条直线时影子长为a,在人到达墙角时影子长为b，我们可以写出人在这两点间时

影子的方程：(x\*H-D\*h)/(x-D)+x，有数学知识可知这个方程在[a,b]之间是凸的，即先增加后减小，

有极值，可以用三分法高效求解，x是人距离墙的水平距离。

虽然能搞出上面的公式，但我还是看不出来它在[a,b]上是凸函数...求解?

\*/

double calc(double x){return (D\*h - x\*H)/(D - x) + x;}

int main()

{

while(cases--) {

scanf("%lf%lf%lf", &H, &h, &D);

double l = 0, r = D\*h/H, mid, midmid;

while(r - l > EPS) {

mid = (l + r) / 2;

midmid = (r + mid) / 2;

if(calc(mid) > calc(midmid)) r = midmid;

else l = mid;

}

printf("%.3f\n", calc(l));

}

return 0;

}