

直接二分出第L大和第R大的数，利用单调性可以在O(nlgANS)内完成，再利用单调性直接把这些数找出来，再把这些数排个序输出，总时间复杂度还是O(nlgAns)

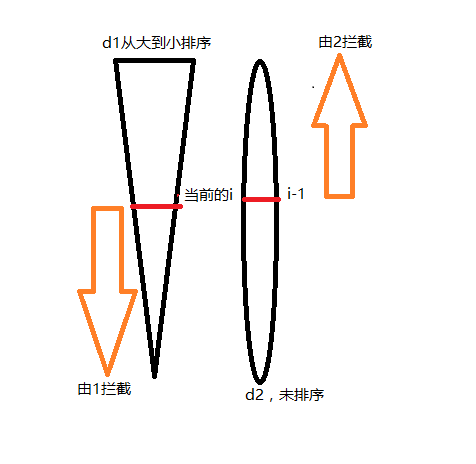
题解：a、b分别排序，二分答案ans，对于ai来说，指针j从b数组的最后一个位置开始向前找到最大的j使得ai+bj<=ans,随着i的增加，指针j是非递增的

直接二分出第L大和第R大的数，利用单调性可以在O(nlgANS)内完成，再利用单调性直接把这些数找出来，再把这些数排个序输出，总时间复杂度还是O(nlgAns)

# NOIP 2010 导弹拦截 排序+贪心

这道题一开始想到了贪心，不过直接贪心肯定不对，想到了先排序。最初的思路是：把所有导弹按照到两个拦截系统较近的距离从大到小排序，之后循环一下，仍然是从两个距离较小的入手，只要令r1或r2满足那个较小的就可以。但这样就华丽丽地挂了3个点，还不明白为什么。看了网友的题解想了好长好长一段时间，才略微地理解了一些。

同样是排序加贪心，不过仅仅按照到第一个拦截系统的距离从大到小排序，之后循环i，答案记录的是i之后（包括i）的导弹全部可以由第一个拦截系统打到（也有可能被第二个打到），i之前（不包括i）的导弹全部可以由第二个拦截系统打到（也有可能被第一个打到），这时的费用和，循环i求一个min，要保证i之后全可以被第一个系统打到很容易，因为导弹是按照第一个拦截系统的距离排序的，保证i之前全可以被第二个打到，需要再维护一个i之前的导弹到第二个拦截系统距离的最大值t。

用一个图形象地解释就是：   
   
下面是代码：

#include <cstdio>#include <algorithm>

using namespace std;

struct gm{

int d1, d2, x, y;

}g[100005];

bool cmp(gm a, gm b){return (a.d1 > b.d1);}

int n, x1, y1, x2, y2, ans = (1<<30), t;

int main(){

scanf("%d %d %d %d %d", &x1, &y1, &x2, &y2, &n);

for(int i = 1; i <= n; i++){

scanf("%d %d", &g[i].x, &g[i].y);

g[i].d1 = (g[i].x-x1)\*(g[i].x-x1) + (g[i].y-y1)\*(g[i].y-y1);

g[i].d2 = (g[i].x-x2)\*(g[i].x-x2) + (g[i].y-y2)\*(g[i].y-y2);

}

sort(g+1, g+n+1, cmp); //记住sort是左闭右开，大部分C++函数都是

for(int i = 1; i <= n; i++){

ans = min(ans, g[i].d1+t);

t = max(t, g[i].d2);

}

printf("%d", min(ans, t));

return 0;

}

问题：有n个数，求从中任取出k个数，使这k个数的最大公约数最大，怎么求啊？(k<=n<=50000,ai<=500000)

从给定的n个数中取出k个数，使得他们的最大公约数最大，求这个最大的公约数

思路：我们可以从大到小枚举公约数，然后统计它的倍数的个数是不是大于等于k，如果是的话那么这个数必然是最大的。

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

#define maxn 100005

int a[maxn];

int main()

{

int n,k,t,x;

scanf("%d",&t);

while(t--)

{scanf("%d%d",&n,&k);

memset(a,0,sizeof(a));

int mmax=0;

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

{

scanf("%d",&x);

a[x]++;

if(x>mmax)

mmax=x;

}

int ans;

bool f=0;

for(int i=mmax;i>=1;i--)

{

int cnt=0;

for(int j=i;j<=mmax;j+=i)

{

cnt+=a[j];

if(cnt>=k)

{

ans=i;

f=1;

break;

}

}

if(f)

break;

}

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

}

return 0;

}