分析：对原始数列进行排序后，首先在这里进行第一次二分搜索，查找中位数mid，然后判断mid是否大了。判断依据如下，计算原数列在i之后，有多少个大于x[i]+mid的数的个数（原始数列里的值比x[i]+mid大，说明该值与x[i]作差形成的新数列里的数比中位数，即mid大），这里求个数就要第二次二分搜索了，可以直接用lower\_bound。如果个数之和小于新数列个数的一半，则说明mid大了。

#include <iostream>

#include <cstdio>

#include <cmath>

#include <cstring>

#include<algorithm>

#define INF 0x3f3f3f3f

#define ll long long

using namespace std;

int n,m;

int a[100010];

bool judge(int val)

{

ll cnt(0);

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

cnt+=a+n-lower\_bound(a+i+1,a+n,a[i]+val);

return cnt>m/2;//cnt计算的是比val大的个数

}

int main()

{

//freopen("input.txt","r",stdin);

while(~scanf("%d",&n))

{

m=n\*(n-1)/2;

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

scanf("%d",&a[i]);

sort(a,a+n);

int l=0,r=\*max\_element(a,a+n);

while(l<=r)

{

int mid=(l+r)>>1;

if(judge(mid))

l=mid+1;

else r=mid-1;

}

printf("%d\n",l-1);

}

return 0;

}