题意：给你2n个点，这个2n个点分别是n个点的x、y坐标，至于哪些是x坐标，些是y坐标不知道，现在问你最小的能把这些点包来的矩形面积是多大

这2n个点将会被划分成2个n个点集合，分别是x坐标和y坐标

面积=(max-min)in set1 \* (max-min) in set2

我们先将2n个点排个序，这2n个点的最大值和最小值有两种情况

1. 都是x坐标或者都是y坐标
2. 一个是x坐标，一个是y坐标

对于①，那么矩形的一边的大小我们已经知道了，就是a[2\*n]-a[1]

那么我们要做到就是让他另一条边最小，也就是要在a[2]到a[2\*n-1]的范围内找到最小的

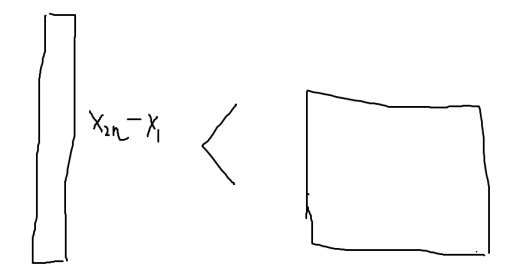
A[i],a[i+1],a[i+1]…..a[i+n]

他们肯定是连续的，因为连续的一段肯定比不连续的小，这是排过序的

对于②，如果最小是x坐标，那么 x坐标的最大值一定>=a[n]，因为x集合的size就是n，同理，如果最大是y坐标，那么y坐标的最小值一定<=a[n+1]，因为y集合的size也是n

这样，这种情况的面积最小值就是(a[n]-a[1])\*(a[2\*n]-a[n+1])

可能会有疑问，为什么会出现①这种情况的，要让面积最小，难道不应该最大和最小分属不同的集合吗

他其实对于这种情况，也就是中间的点（比如a[n]附近的）特别密集，而中间的点和边缘的点(a[1],a[2n])相距又很大时，这时②的情况的话，两条边都会比较大，而①虽然一条边大，很另一条边因为中间的点很密，靠的很近而很小，这样面积就有可能①<②

也可以把它看成染色

2C就是2n个数，染n个红色，n个蓝色，求(红最大-红最小)\*(蓝最大-蓝最小)的最小

考虑先排序，然后考虑两端是否染同一个颜色

如果不染同一个颜色，那最优的肯定是左边n个红右边n个蓝

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int INF=1000000007;

ll a[1000010];

int main()

{

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

int n;

cin>>n;

for(int i=0;i<2\*n;i++)

cin>>a[i];

sort(a,a+2\*n);

ll x=a[n-1]-a[0];

ll y=a[2\*n-1]-a[n];

ll ans=INF;

int r;

for(int l=1;l<n;l++)

{

r=l+n-1;

if(ans>a[r]-a[l])

ans=a[r]-a[l];

}

ans\*=a[2\*n-1]-a[0];

if(ans>x\*y)ans=x\*y;

cout<<ans<<endl;

return 0;

}