**Description**

恰逢 H 国国庆，国王邀请 n 位大臣来玩一个有奖游戏。首先，他让每个大臣在左、右手上面分别写下一个整数，国王自己也在左、右手上各写一个整数。然后，让这 n 位大臣排成一排，国王站在队伍的最前面。排好队后，所有的大臣都会获得国王奖赏的若干金币，每位大臣获得的金币数分别是：排在该大臣前面的所有人的左手上的数的乘积除以他自己右手上的数，然后向下取整得到的结果。   
国王不希望某一个大臣获得特别多的奖赏，所以他想请你帮他重新安排一下队伍的顺序，   
使得获得奖赏最多的大臣，所获奖赏尽可能的少。注意，国王的位置始终在队伍的最前面。

**Input**

第一行包含一个整数 n，表示大臣的人数。   
第二行包含两个整数 a 和 b，之间用一个空格隔开，分别表示国王左手和右手上的整数。 接下来 n 行，每行包含两个整数 a 和 b，之间用一个空格隔开，分别表示每个大臣左手和右手上的整数。

**Output**

输出只有一行，包含一个整数，表示重新排列后的队伍中获奖赏最多的大臣所获得的金币数。

**Sample Input**

3

1 1

2 3

7 4

4 6

**Sample Output**

2

**Hint**

【输入输出样例说明】   
按1、2、3号大臣这样排列队伍，获得奖赏最多的大臣所获得金币数为2；   
按1、3、2这样排列队伍，获得奖赏最多的大臣所获得金币数为2；   
按2、1、3这样排列队伍，获得奖赏最多的大臣所获得金币数为2；   
按2、3、1这样排列队伍，获得奖赏最多的大臣所获得金币数为9；   
按3、1、2这样排列队伍，获得奖赏最多的大臣所获得金币数为2；   
按3、2、1这样排列队伍，获得奖赏最多的大臣所获得金币数为9。   
因此，奖赏最多的大臣最少获得 2 个金币，答案输出 2。   
  
对于 20%的数据，有 1≤ n≤ 10，0 < a、b < 8；   
对于 40%的数据，有 1≤ n≤20，0 < a、b < 8；   
对于 60%的数据，有 1≤ n≤100； 对于 60%的数据，保证答案不超过 10^9；   
对于 100%的数据，有 1 ≤ n ≤1,000，0 < a、b < 10000。

   看到这道题的使得获得奖赏最多的大臣，所获奖赏尽可能的少，我的第一想法是二分答案。但是又仔细想了想，答案不具有单调性，二分是行不通的。

    再看看这道题，我感觉可以用贪心来做。也就是说先用贪心确定一种最优的方案，再在这个排列队伍中求得答案。

    可以这样思考，相邻两个的人交换对于前面的人答案没影响（这不是废话么），而且对于后面的人答案也没有影响（这里很关键）。也就是说相邻两人的位置交换只会对这两个人产生影响。那么我们就先考虑这两个人。

        设这两个人分别为i和i+1。左手数字为a[i]和a[i+1]，右手数字为b[i]和b[i+1]。两人的金币数为w[i]和w[i+1]。

        记P[i]=a[1]\*a[2]\*a[3]\*...\*a[i]

        可得：

            w[i]=P[i-1]/b[i];

            w[i+1]=P[i]/b[i+1];

        又P[i]=P[i-1]\*a[i]

        那么 w[i+1]=P[i-1]\*a[i]/b[i+1]=w[i]\*a[i]\*b[i]/b[i+1]

    不难看出，在这个相邻的二元组中，前面的数不受后面的影响，而后面的金币数决定于w[i],a[i],b[i]。

    推广到整个排队方案，前面的金币数和a[i],b[i]都会影响后面的答案。贪心原则便出来了：按a[i]\*b[i]为关键字从小到大排序，相同的顺序无所谓。最后再扫一遍，算出答案即可。

    此题需要注意一点：乘除法都要写高精度，且答案有10000多位。

附一个不用高精度的60%数据的代码

#include <iostream>

#include <cstring>

#include <cstdio>

#include <algorithm>

using namespace std;

const int maxn = 1015;

int n,m;

long long ans = 0,ccj = 0;

struct edge

{

int l,r;

long long cnt,cj;

}num[maxn],gw;

bool cmp(edge a,edge b)

{

return a.cj < b.cj;

}

int main()

{

cin>>n;

cin>>gw.l>>gw.r;

gw.cj = gw.l\*gw.r;

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

{

cin>>num[i].l>>num[i].r;

num[i].cj = num[i].l \* num[i].r;

}

sort(num + 1,num + 1 + n,cmp);

num[0].l = gw.l;

num[0].r = gw.r;

num[0].cj = gw.cj;

ccj = num[0].l;

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

{

num[i].cnt=ccj/num[i].r;

ccj\*=num[i].l;

}

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

ans = max(ans,num[i].cnt);

cout<<ans<<endl;

return 0;

}