题目描述

Perket 是一种流行的美食。为了做好 Perket，厨师必须谨慎选择食材，以在保持传统风味的同时尽可能获得最全面的味道。你有 n 种可支配的配料。对于每一种配料，我们知道它们各自的酸度 s 和苦度 b。当我们添加配料时，总的酸度为每一种配料的酸度总乘积；总的苦度为每一种配料的苦度的总和。

众所周知，美食应该做到口感适中，所以我们希望选取配料，以使得酸度和苦度的绝对差最小。

另外，我们必须添加至少一种配料，因为没有任何食物以水为配料的。

输入格式

第一行一个整数 n，表示可供选用的食材种类数。

接下来 n 行，每行 2 个整数 si和 bi，表示第 i种食材的酸度和苦度。

输出格式

一行一个整数，表示可能的总酸度和总苦度的最小绝对差。

输入样例

```
1
3 10
```

输出样例

```
7
```

解析

这也是一道子集问题，因此我们同样可以使用二进制方法枚举全部的可能性，来进行求解。

编码

```cpp
#include<bits/stdc++.h>

using namespace std;

//定义食材的最大数量
const int maxn = 15;
//定义食材酸度和苦度的结构体
struct node {
    int s, b;
} s[maxn];
```

```
//食材的数量
int n;
//因为要求小的差值，所以最初的时刻要初始化为最大值
int sum = 0x3f3f3f3f;

int main() {
    scanf("%d", &n);
    //读入每个食材的酸度和苦度
    for (int i = 1; i <= n; i++) {
        scanf("%d%d", &s[i].s, &s[i].b);
    }
    //二进制变化
    int u = 1 << n;
    //注意t的最小值就是1，也就是说至少要选择一种材料
    for (int t = u - 1; t >= 1; t--) {
        //临时的和
        int tempSum = 1;
        //临时的积
        int tempProduct = 0;
        //j右移到n，保证最小值为1
        for (int j = 1; j <= n; j++) {
            int temp = u >> j;
            //如果当前位置存在数字
            if (t & temp) {
                tempSum *= s[j].s;
                tempProduct += s[j].b;
            }
        }
        //比较最小的值
        sum = min(sum, abs(tempSum - tempProduct));
    }

    printf("%d\n", sum);
    return 0;
}
```