P3743 kotori的设备

题目描述

第 i 个设备每秒消耗ai个单位能量。能量的使用是连续的，也就是说能量不是某时刻突然消耗的，而是匀速消耗。也就是说，对于任意实数 ,在 k 秒内消耗的能量均为k*ai 单位。在开始的时候第 i 个设备里存储着bi个单位能量。

同时 kotori 又有一个可以给任意一个设备充电的充电宝，每秒可以给接通的设备充能p 个单位，充能也是连续的，不再赘述。你可以在任意时间给任意一个设备充能，从一个设备切换到另一个设备的时间忽略不计。

kotori 想把这些设备一起使用，直到其中有设备能量降为 0。所以 kotori 想知道，

在充电器的作用下，她最多能将这些设备一起使用多久。

输入格式

第一行给出两个整数 n,p。

接下来 n 行，每行表示一个设备，给出两个整数，分别是这个设备的ai 和 bi

输出格式

如果 kotori 可以无限使用这些设备，输出-1。

否则输出 kotori 在其中一个设备能量降为 0 之前最多能使用多久

设你的答案为 a,标准答案为 b,只有当 a,b 满足 $\frac{|a-b|}{\max(1,b)} \leq 10^{-4}$ 候，你能得到本测试点的满分。

输入样例

```
2 1
2 2
2 1000
```

输出样例

```
2.0000000000
```

解析

二分法需要考虑的第一个问题是：我们在搜索什么？答案很明显是最长的使用时间。

我们可以得到如下公式：

最长的使用时间 * 设备输出电量 = 设备现有电量 + 单位时间内的充电量 * 充电时间

又因为题目中描述可以在任意时刻为任意设备充电，因此极限情况就是我们可以从最初的那一刻起为设备开始充电，这样才能使设备的使用率达到最大，即：**“充电时间”**只要小于等于枚举的**“最长的使用时间”**，我们就认为该设备能够正常工作。

此外，还有一种极限情况，即**“最长的使用时间”**满足全部设备的**“充电时间”**，但是充电设备只有一个，我们只能在全部的设备中选择一个进行充电。因此，当每个**“充电时间”**累加之后，如果其和大于**“最长的使用时间”**那么也是无效的。

最后，这道题目中没有给出上限值，我们只能从较大的数值开始枚举，在这里我给出的右边界是1e10，即10亿规模。如果得出的答案与上限值的差值小于eps，我们就认为其充电能力能够源源不断的提供电能。

编码

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

using namespace std;

//设备数量和固定充电值
double n, p;

const double eps = 1e-6; //定义我们计算的精度
const double maxPower = 1e10; //浮点数的右边界最大值为1e10
//每秒消耗的电能数组
double a[100001];
//每秒能充入的电能数组
double b[100001];

//检测当前设备是否能持续x的时间
bool check(double x) {
    double totalTime = 0;
    for (int i = 0; i < n; ++i) {
        double useTime = b[i] / a[i];
        //电量持续时间超过x
        if (useTime >= x) {
            continue;
        } else {
            //所需要持续的时间
          double restTime = x - useTime;
            //所需要的能量
          double needPower = restTime * a[i];
            //所需要的充电时间
          double needTime = needPower / p;
```

```
            //某个设备的充电时间小于等于所需时间，可以继续
            if (needTime <= x) {
                totalTime += needTime;
                continue;
            }
            //某个设备的充电时间大于所需时间，直接判定失败
            else {
                return false;
            }
        }
    }
    //只有全部设备的所需的充电时间之和小于给定时间，
    //才能够持续给定的时间
    return totalTime <= x;
}

//执行二分搜索
double BinarySearch(double l, double r) {
    double mid;
    //通过与eps作比较来判断是否找到根
    while (r - l > eps) {
        //计算中值
        mid = l + (r - l) / 2;
        bool res = check(mid);
        //能够支撑，尝试用更长的时间
        if (res) {
            l = mid;
        }
            //不能支撑，降低时间
        else {
            r = mid;
        }
    }
    //返回更大的持续时间
    return r;
}

int main(int argc, char *argv[]) {
    cin >> n >> p;
    for (int i = 0; i < n; ++i) {
        cin >> a[i] >> b[i];
    }
    double res = BinarySearch(0, maxPower);
    //如果得到的最大值与边界相近，则认为是可以无限使用的
    if (maxPower - res <= eps) {
        printf("-1");
    } else {
        printf("%lf", res);
    }
```

```
    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。