篝火晚会

(fire.pas/c/cpp)

【问题描述】

佳佳刚进高中，在军训的时候，由于佳佳吃苦耐劳，很快得到了教官的赏识，成为了“小教官”。在军训结束的那天晚上，佳佳被命令组织同学们进行篝火晚会。一共有 n 个同学，编号从 1 到 n。一开始，同学们按照 1，2，……，n 的顺序坐成一圈，而实际上每个人都有两个最希望相邻的同学。如何下命令调整同学的次序，形成新的一个圈，使之符合同学们的意愿，成为摆在佳佳面前的一大难题。

佳佳可向同学们下达命令，每一个命令的形式如下：

(b_1, b_2,... b_{m-1}, b_m)

这里 m 的值是由佳佳决定的，每次命令 m 的值都可以不同。这个命令的作用是移动编号是 b_1，b_2，…… b_{m-1}，b_m的这 m 个同学的位置。要求 b_1换到 b_2的位置上，b_2换到 b_3的位置上，……，要求 b_m换到 b_1的位置上。

执行每个命令都需要一些代价。我们假定如果一个命令要移动 m 个人的位置，那么这个命令的代价就是 m。我们需要佳佳用最少的总代价实现同学们的意愿，你能帮助佳佳吗？

【输入文件】

输入文件 **fire.in** 的第一行是一个整数 n（3 <= n <= 50000），表示一共有 n 个同学。其后 n 行每行包括两个不同的正整数，以一个空格隔开，分别表示编号是 1 的同学最希望相邻的两个同学的编号，编号是 2 的同学最希望相邻的两个同学的编号，……，编号是 n 的同学最希望相邻的两个同学的编号。

【输出文件】

输出文件 **fire.out** 包括一行，这一行只包含一个整数，为最小的总代价。如果无论怎么调整都不能符合每个同学的愿望，则输出-1。

【样例输入】

```
4
3 4
4 3
1 2
1 2
```

【样例输出】

```
2
```

【数据规模】

对于 30%的数据，n <= 1000；

对于全部的数据，n <= 50000。

【问题分析】

在考虑问题的解决方法之前，我们先考虑问题的最终状态：它是一个排列。很显然，我们首先可以根据题目给出的信息，用扫描的办法构造出一组可行排列：从第 1 个人开始，先任意访问一个与之相邻的人 A，接着，再从 A 开始，依次访问下去，最终如果回到第 1 个人，则说明问题有解，否则问题无解。

由于本问题人都是围成一个圆圈的，所以考虑开始状态是 $(1 \quad 2 \quad \ldots \quad n)$，我们首先求出的可行目标状态为 $(b_1 \quad b_2 \quad \ldots \quad b_n)$，则 $(b_n \quad b_{n-1} \quad \ldots \quad b_1)$ 也是可行的目标状态，并且他们旋转所形成的状态也是可行的目标状态。而问题的解决，就是在这些所有可行的目标状态中，找出所需交换次数最少的。

下面考虑从初状态 $(1 \quad 2 \quad \ldots \quad n)$ 到目标状态 $(b_1 \quad b_2 \quad \ldots \quad b_n)$ 的最少交换次数：

(1) 我们可以找到交换次数的最小值：对于 $\begin{pmatrix} 1 & 2 & \ldots & n \\ b_1 & b_2 & \ldots & b_n \end{pmatrix}$，设有 p 个元素存在 $b_i = i$。显然，这 p 个元素是不需要调整的。而剩下的所有元素，每个至少调整一次，则交换次数的最小值为 $N - p$。

(2) 是否一定存在交换次数为 $N - p$ 的解答呢？答案是肯定的：对于置换群 $\begin{pmatrix} 1 & 2 & \ldots & n \\ b_1 & b_2 & \ldots & b_n \end{pmatrix}$，由于交换的次序任意，每一个循环 $\begin{pmatrix} b_1 & b_2 & \ldots & b_m \\ c_1 & c_2 & \ldots & c_n \end{pmatrix}$ 中，我们按照 $(1 \quad 2 \quad \ldots \quad m)$ 的顺序，只需修改一次，能将循环重置成任意想要的顺序，而这个操作消耗题目中的代价为 m。于是对于每一个循环都如此操作，总的操作代价就是 $N - p$。

综上所述，要想构造可行解，由(1)知交换次数不得少于 $N - p$，由(2)知数目为 $N - p$ 的方案是一定存在的，于是最优解即为 $N - p$。

按照这个思路，我们需要对每一组可行解进行一次扫描，每一次扫描的时间复杂度为 $O(N)$，共进行 $2N$ 次，其时间复杂度为 $O(N^2)$，对于 $N = 50000$，是不能在规定时限内得到解的。通过观察我们发现：对于 $\begin{pmatrix} 1 & 2 & \ldots & n \\ b_1 & b_2 & \ldots & b_n \end{pmatrix}$，无论如何旋转，$b_i$ 与 i 的相对位置是不会改变的。所以，我们只要首先计算出 b_i 与 i 的相对位置，将它们放入 Hash 表，最后找出 Hash 表中的最大值（对应了上文中 p 的最大值），就可以在 $O(N)$ 的时间内统计出 $N - p$ 的最小值，也就是最优解，问题解决。

【代码清单】

```
#include <cstdio>
#include <cstring>
```

```
const int MaxN = 50000 + 100;

int N, Sum = 0;
int Left[MaxN], Right[MaxN], P[MaxN], Hash[MaxN];
bool visited[MaxN];

void init() {
    freopen("fire.in", "r", stdin);
    freopen("fire.out", "w", stdout);

    scanf("%d", &N);
    for (int i = 1; i <= N; i ++)
        scanf("%d%d", &Left[i], &Right[i]);
}

bool proc() { //求出一组可行解
    int Cnt = 1; visited[1] = true;
    for (int i = 1; i < N; i ++) {
        P[i] = Cnt;
        if (!visited[Left[Cnt]]) Cnt = Left[Cnt];
        else Cnt = Right[Cnt];
        if (visited[Cnt]) return false;
        visited[Cnt] = true;
    }
    P[N] = Cnt;
    return true;
}

int main() {
    int i;

    init();
    if (!proc()) printf("-1\n");
    else {
        int Max = 0;
        for (i = 1; i <= N; i ++)
            if (P[i] <= i) Hash[i - P[i]] ++;
            else Hash[N + i - P[i]] ++; //计算与递增排列的位移
        for (i = 0; i <= N; i ++)
            if (Hash[i] > Max) Max = Hash[i]; //统计最优解

        memset(Hash, 0, sizeof(Hash));
        for (i = 1; i <= N; i ++)
            if (P[N - i + 1] <= i) Hash[i - P[N - i + 1]] ++;
```

```
            else Hash[N + i - P[N - i + 1]] ++; //计算与逆向排列的位移
        for (i = 0; i <= N; i ++)
            if (Hash[i] > Max) Max = Hash[i]; //统计最优解
        printf("%d\n", N - Max);
    }
    return 0;
}
```

【小结】

对拿到手不会做的试题，要大胆猜想，敢于分析，最终才能把它解决。