

红黑图

知识点枚举

时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

众所周知 **红黑树**^Q 是一种平衡树，它最突出的特性就是不能有两个相邻的红色节点。

那我们定义一个红黑图，也就是一张无向图中，每个节点可能有红黑两种颜色，但我们必须保证没有两个相邻的红色节点。

现在给出一张未染色的图，只能染红黑两色，问总共有多少种染色方案使得它成为一个红黑图。

输入描述：

第一行两个数字 n m ，表示图中有 n 个节点和 m 条边。

接下来共计 m 行，每行两个数字 s t ，表示一条连接节点 s 和节点 t 的边，节点编号为 $[0, n)$ 。

输出描述：

一个数字表示总的染色方案数。

补充说明：

$0 < n < 15$

$0 \leq m \leq n * 3$

$0 \leq s, t < n$

不保证图连通

保证没有重边和自环

示例1

输入：

3 3
0 1
0 2
1 2

输出：

4

示例2

输入：

4 3
0 1
1 2
2 3

输出：

8

示例3

输入：

4 3
0 1
0 2
1 2

输出：

8

解题思路：

求出红黑图所有的配色方案。方案总个数 2^n 。
遍历配色方案是否符合无相邻红色，如不符合则从总数中-1。

```
public class Main{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
```

```

int n = sc.nextInt();
int m = sc.nextInt();

List<int[]> list = new ArrayList<>();    //相连的节点数组集合
for(int i=0; i<m; i++){
    int[] points = new int[2];
    points[0] = sc.nextInt();
    points[1] = sc.nextInt();
    list.add(points);
}

int total = (int) Math.pow(2, n);    //n 个节点有 2^n 个红黑搭配
int res = total;
for(int i=0; i < total; i++){

    int temp = i;
    int[] ints = new int[n];    //红黑搭配的情况（0 表示红，1 表示黑）
    for(int j=0; j<n; j++){
        ints[j] = temp % 2;
        temp /= 2;
    }

    for(int[] points : list){
        if(ints[points[0]] == 0 && ints[points[1]] == 0){    //相连的节点都是红色，表
示不符合
            res--;
            break;    //任意一个相连不符则表示此情况不符合
        }
    }
}

System.out.println(res);
}
}

```