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:

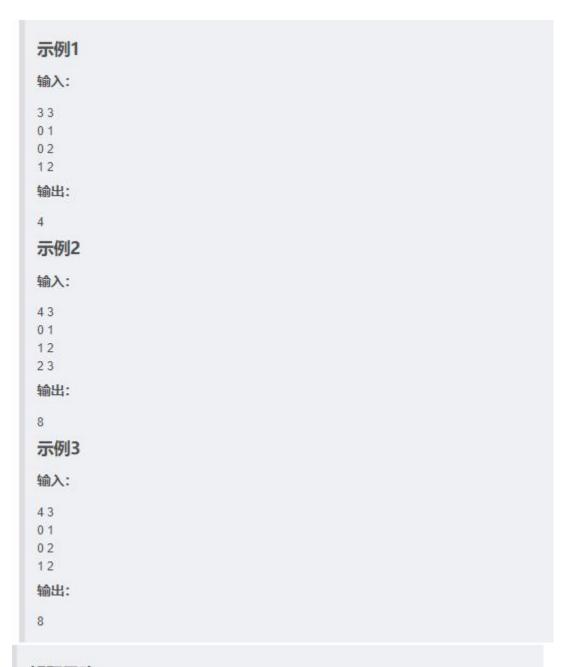
0 < n < 15

0 <= m <= n * 3

0 <= s, t < n

不保证图连通

保证没有重边和自环



解题思路:

求出红黑图所有的配色方案。方案总个数2ⁿ。 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++){</pre>
             int temp = i;
                                     //红黑搭配的情况(0表示红,1表示黑)
             int[] ints = new int[n];
             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);
    }
}
```