207 课程表

Label：图
你这个学期必须选修 numCourses 门课程，记为 0 到 numCourses - 1 。
在选修某些课程之前需要一些先修课程。 先修课程按数组 prerequisites 给出，其中 prerequisites[i] = [ai, bi] ，表示如果要学习课程 ai 则 必须 先学习课程 bi 。

例如，先修课程对 [0, 1] 表示：想要学习课程 0 ，你需要先完成课程 1 。
请你判断是否可能完成所有课程的学习？如果可以，返回 true ；否则，返回 false 。

输入：numCourses = 2, prerequisites = [[1,0]]
输出：true
解释：总共有 2 门课程。学习课程 1 之前，你需要完成课程 0 。这是可能的。

输入：numCourses = 2, prerequisites = [[1,0],[0,1]]
输出：false
解释：总共有 2 门课程。学习课程 1 之前，你需要先完成•课程 0 ；并且学习课程 0 之前，你还应先完成课程 1 。这是不可能的。

- DFS

```
class Solution {
    public boolean canFinish(int numCourses, int[][] prerequisites) {
        List<List<Integer>> adjacency = new ArrayList<>();  // 记录每门先修课的后修课

        for(int i = 0; i < numCourses; i++)
            adjacency.add(new ArrayList<>());

        int[] flags = new int[numCourses]; // 标记可以修

        for(int[] cp : prerequisites)
            adjacency.get(cp[1]).add(cp[0]);

        for(int i = 0; i < numCourses; i++)
            if(!dfs(adjacency, flags, i))  // 有环
                return false;

        return true;
    }
    private boolean dfs(List<List<Integer>> adjacency, int[] flags, int i) {
        if(flags[i] == 1)  return false; // 依赖于已经修过的，有环


        if(flags[i] == -1)  return true; // 无环

        flags[i] = 1;
        for(Integer j : adjacency.get(i))
            if(!dfs(adjacency, flags, j)) return false;
        flags[i] = -1;

        return true;
    }
}
```