一、结点的设计：

角色，而不是用户，作为结点

人事变动频繁，但流程结构很难变。

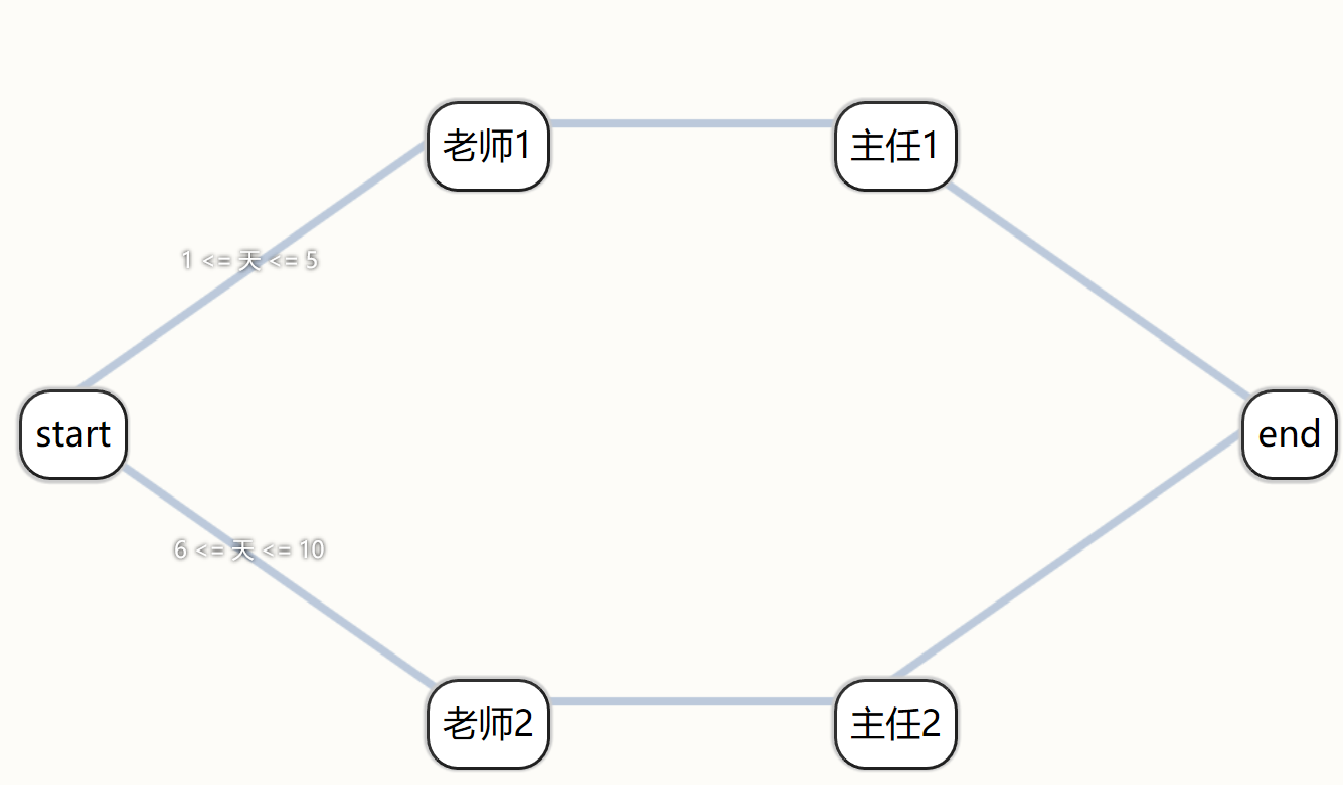
为角色分配用户，随时调整担任角色的用户，更加灵活

二、创建申请后得到申请的结构图

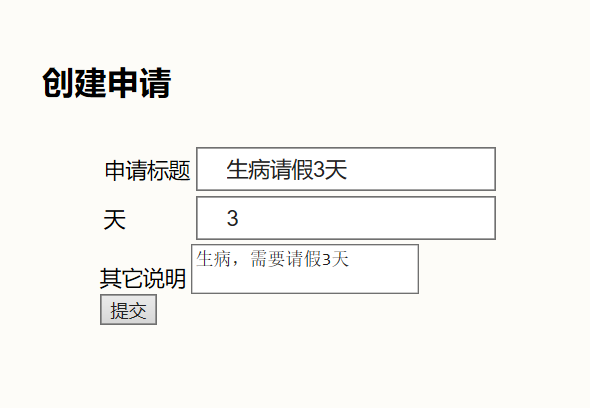
向用户隐藏不需要的图结构

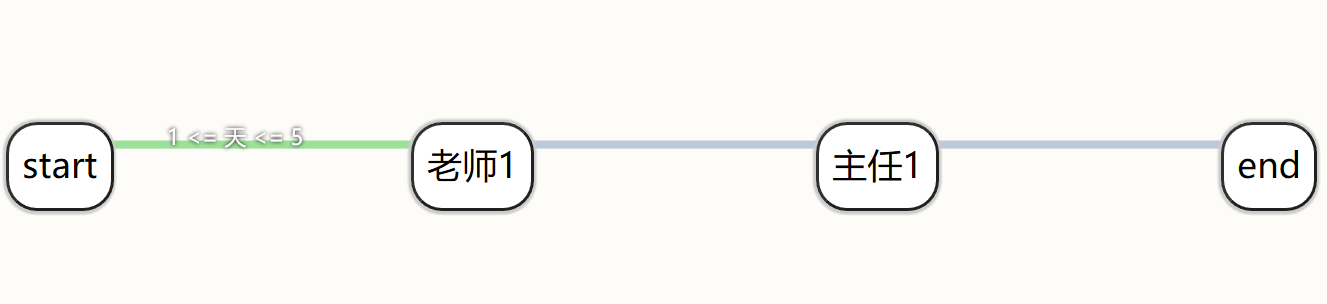
填写申请表后，根据其填写的条件变量值的不同，申请对应的结构图也不同。例如：

对于这样一个申请：

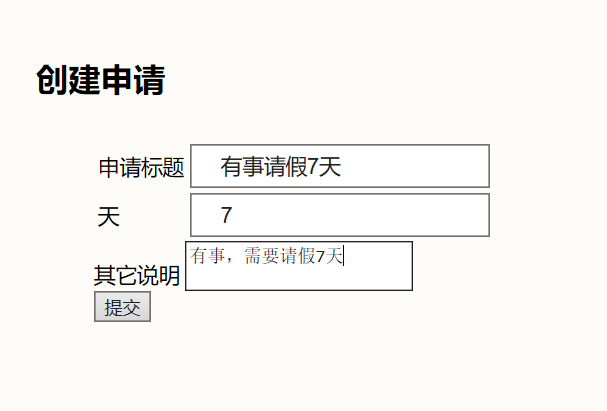


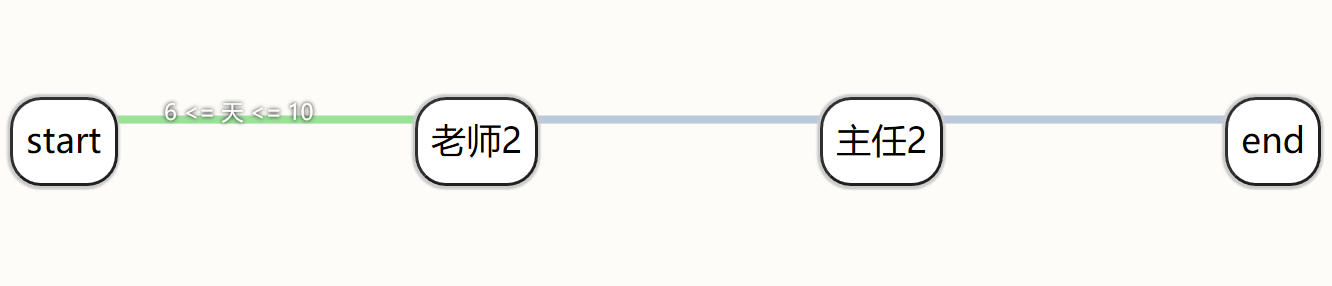
如果请假3天，那么其申请对应的流程图应该为





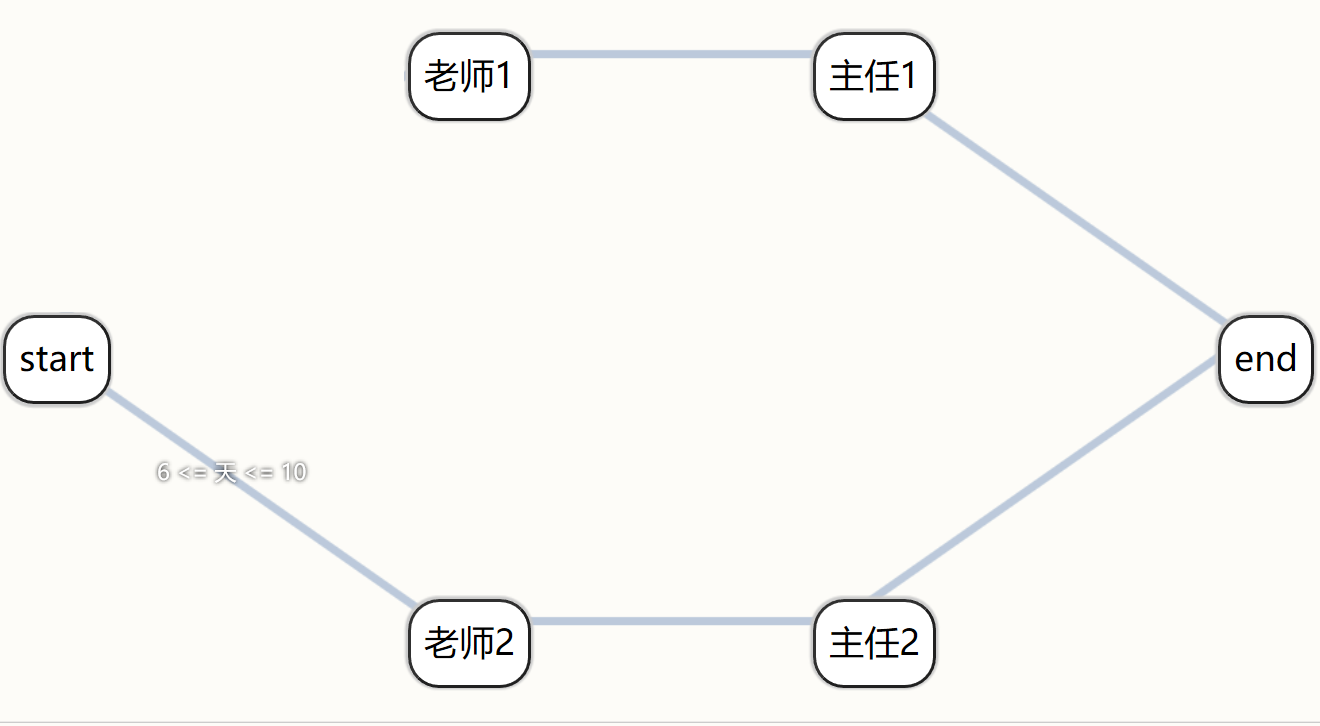
而如果请假7天，那么其申请对应的流程图应该为





三、后端具体实现：

读入对应流程的结构图，根据所有传输过来的条件变量的值，将不符合条件变量的边删除。但是仅仅这样是不够的，因为会出现这种情况：



使用广度优先搜索，只有既能被起点访问到又能被终点访问到的点才不会被删除。

四、审批后，后端的处理

若审批通过：

1. 在数据库中，将当前审批人在申请结构图中的所有出边的结果更新为“通过”。
2. 查询当前审批人的所有出边的邻接点。对于每一个邻接点，查询其所有入边是否全部通过。若全部通过，则发送邮件提醒该审批人。若邻接点是end节点，且end节点的所有入边均通过，则发送邮件提醒申请人，申请已经通过。

若审批不通过：

1. 在数据库中，将当前审批人在申请结构图中的所有出边的结果更新为“不通过”。
2. 将该申请的结果更新为“不通过”。