1. 合并时间区间（建议时间复杂度 O(n)）

给定一个**按开始时间从小到大排序**的时间区间集合，请将重叠的区间合并。时间区间集合用一个二维数组表示，二维数组的每一行表示一个时间区间（**闭区间**），其中 0 位置元素表示时间区间开始，1 位置元素表示时间区间结束。

例 1：输入：[[1, 3], [2, 6], [8, 10], [15, 18]]
返回：[[1, 6], [8, 10], [15, 18]]
解释：时间区间 [1, 3] 和 [2, 6] 有部分重叠，合并之后为 [1, 6]

例 2：输入：[[1, 4], [4, 5]]
返回：[[1, 5]]
解释：时间区间[1，4] 和 [4，5]重叠了一个时间点，合并之后为 [1，5]

需要实现的方法原型：`int[][] merge(int[][] intervals)`

2. 缩写校验（建议时间复杂度 O(n)）

给定一个**非空**字符串 s 和一个缩写 abbr，请校验它们是否匹配。

假设字符串中只包含小写字母，缩写中只包含小写字母和数字。缩写中的数字表示其缩略的字符数；连续多位数字表示一个多位数。

例如，字符串 “word” 的缩写有且仅有以下这些：[“word", "1ord", "w1rd", "wo1d", "wor1", "2rd", "w2d", "wo2", "1o1d", "1or1", "w1r1", "1o2", "2r1", "3d", "w3", “4"]。

例 1：输入：s = “internationalization", abbr = "i12iz4n"
返回：true
解释：abbr 中的 12 表示有十二个字符被缩略了。

例 2：输入：s = “apple", abbr = “a2e"
返回：false

需要实现的方法原型：`boolean valid(String word, String abbr)`

3. 最小惩罚

给定一个 **无向图** 包含 N 个节点和 M 条边, 每条边 M_i 的代价为 C_i。图中一条路径的惩罚是指对该路径上所有边的代价 C_i 执行位运算或（bitwise OR）操作得到的。假如一条路径上包含了边 M_1，M_2，M_3 … … ，M_k，那么对应的惩罚就是 C_1 OR C_2 OR C_3 OR ... OR C_k。（OR代表位运算或，即 “|” ）

问题：给定图上两个节点 start 和 end，找到从 start 到 end 的所有路径中惩罚值最小的路径，对应的最小惩罚值作为结果返回。如果路径不存在就返回 -1。

注意：任意两个节点之间最多存在一条边，图中可能存在有环路。

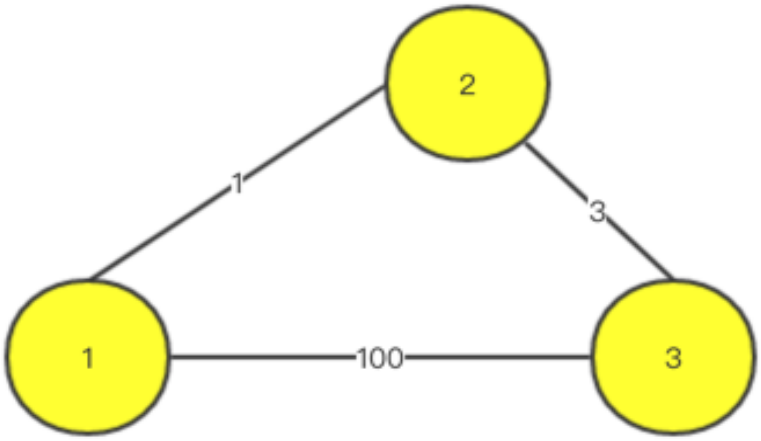
需要实现的方法原型：

```
int minPath(int n, int[][] edges, int start, int end)
```

参数含义：
n：节点总数；节点编号从 1 开始，一直到 n，共有 n 个；
edges：无向图的边；edges[i] 表示边M_i，其中 edges[i][0] 和 edges[i][1] 是M_i的两个节点的编号，edges[i][2] 是M_i对应的代价 C_i；
start 和 end：路径的开始和结束节点编号

限制条件: 1 <= n <=1000
1 <= edges.length <= 10000
1 <= C_i <=1024

例：edges = [[1，2，1]，[2，3，3]，[1，3，100]]，对应的图如下：



当 start = 1，end = 3 时，其最小惩罚路径是 1->2->3, C(1,2)=1并且C(2,3)=3, 对应的惩罚值为 1 | 3 = 3。