GO-图-某通信网络中有 N 个网络结点
题目描述：
某通信网络中有 N 个网络结点，用 1 到 N 进行标识。网络通过一个有向无环图表示，其中图的边的值表示结点之间的消息传递时延。
现给定相连节点之间的时延列表 times[i]={u,v,w}，其中 u 表示源结点，v 表示目的结点，w 表示 u 和 v 之间的消息传递时延。请计算给定源结点到目的结点的最小传输时延，如果目的结点不可达，返回-1。

注：
1、N 的取值范围为[1,100]；
2、时延列表 times 的长度不超过 6000，且 1 <= u,v <= N, 0 <= w <= 100；
输入描述：
输入的第一行为两个正整数，分别表示网络结点的个数 N，以及时延列表的长度 M，用空格分隔；
接下来的 M 行为两个结点间的时延列表[u v w]；
输入的最后一行为两个正整数 u 和 v，分别表示源结点和目的结点；
输出描述：
输出一个整数，表示源结点到目的结点的最小时延。
补充说明：
示例 1
输入：
3 3
1 2 11
2 3 13
1 3 50
1 3
输出：
24
说明：
1->3 的时延是 50，1->2->3 时延是 11+13=24，所以 1 到 3 的最小时延是 24；

```
package main

import (
	"fmt"
)

func main() {
	n := 0
	m := 0
	source := 0
	dst := 0
	fmt.Scan(&n,&m)
	nodes:=[][3]int{}
	for i:=0 ; i<m ; i++ {
```

```
		cur := [3]int{}
		fmt.Scan(&cur[0],&cur[1],&cur[2])
		nodes=append(nodes, cur)
	}
	fmt.Scan(&source,&dst)
	min,err:= findpath(source,dst,nodes)
	if !err {
		min = -1
	}
	fmt.Println(min)
}

func findpath(source int,dst int,nodes [][3]int) (int,bool){
	min := 0
	find := true
	if source == dst {
		return 0,true
	} else {
	for i:=0 ; i<len(nodes) ; i++ {
		if nodes[i][0] == source {
				cur,err := findpath(nodes[i][1],dst,nodes)
				cur= cur + nodes[i][2]
				if (cur < min || min == 0) && err {
					min =cur
				}
			}
		}
	if min==0 {
		find = false
	}
	}
	return min,find
}
```