A

题意：

给出一幅图，图上是由字母组成的点，字母与字母之间有一些边。求最小生成树。

直接套Kruskal算法即可，用结构体存储边，按边长排序，排序后，用并查集不断合并不在同一个“集合”的点，同时计算边长。详情见代码。

可能输入有点不容易看明白。

第一行输入字母数n（点数）。

然后接下来的n-1行，每行依次是：该点编号——边的数目——边的数目个边。

例如样例中的

```
A 2 B 12 I 25
```

表示结点A有两条边，连向B的边的边长为12，连向I的边的边长为25。

B

题意：

给出一幅图n个点，然后是一个n*n的矩阵。

信息是：（以样例为例）

```
0 990 692
990 0 179
692 179 0
```

第一行：第一个点到第一个点的距离为0，到第二个点的距离为990，到第三个点的距离为692

第二行：第二个点到第一个点的距离为990，到第二个点的距离为0，到第三个点的距离为179

第三行：第三个点到第一个点的距离为692，到第二个点的距离为179，到第三个点的距离为0

然后输入m，表示接下来有m条边是已经有的（意思就是说它的花销不用加进最小生成树的边长之中）。接下来m行表示已有的边的两个端点。

```
1
1 2
```

以样例为例，就是说有一条边是已有的，然后是1和2之间的边。

思路：

考虑到Kruskal算法得到最小生成树的过程，是通过给边的长度排序再进行操作。

那么这道题中，应该能想到已有的边是肯定要放到最小生成树中，并且不计其花销的，也就是说在优先考虑使用已有的边的同时，不计其“花销”。

所以，读入整个矩阵后，把已有的边的长度设置为0，再跑最小生成树即可。

C

题意：

给出一幅有向图。让求一条从编号为1的点到编号为n的点的路径，使得这条路径上最小的边的权值“allowed weight”最大。

思路：

从Dijkstra变形而来。

让我们回忆一下Dijkstra——用d数组，其中d[i]表示从源点到该点的距离。执行“更新”操作或者说“松弛”操作的条件是——当上一个点i与该点j的距离g[i][j]加上d[i]后小于d[j]。

类比于这道题的条件——我们用d[i]表示从源点到该点的路径上最小的allowed weight，执行松弛操作的条件是——当上一个点i与该点j之间的allowed weight与d[i]两者间的较小值大于d[j]。

这里需要注意两点：相对于最短路求“最小值”，这里是求最大值，所以dijkstra函数中<的方向要改一下；其次是，若判断到j的时候发现d[j]是第一次赋值，则直接赋予min(d[u],g[u][j])，而跳过对d[u]和d[j]两者大小的判断。
详情见代码。

此外还可以用最大生成树（只是把最小生成树中对边的排序由从小到大换成从大到小）的想法来写。同样的每次判断到两个点不属于同一集合时，合并之，合并后判断1和n是否处于同一集合中，若是，则当前对应边即为所求。（因为边是从大到小排序，所以最后一条加入后使得1和n连通的边一定是这些边中边权最小的）

D
题意：
给出排好的多米诺骨牌，每次推到第一块。求最后一块多米诺骨牌倒下的时间。
思路：
有两种情况：一种是最后一块倒下的是题目给出的“结点”，另一种是两个结点之间的某一块。那么我们可以用Dijkstra算法求出每个“结点”倒下的时间（最短路），然后——
答案要么是结点倒下的时间中的最大值，要么是(d[i]+d[j]+g[i][j])/2（骨牌i和骨牌j之间的最后一个骨牌倒下的时间）——这里枚举所有边即可（代码中用了Edge结构体保存边，方便最后遍历）。
注意当n为1，m为0的时候，是否能正确输出。

E
题意：
。。。其实这题看懂题意好像比做出这道题要难。
给出n个点（经纪人）和一些边（信息传递关系），注意边是有向边，求消息传遍整个“关系网”所需要的最短的时间（是不是跟D有点像？），并输出对应开始传出消息的人的编号。如果无论如何都有人得不到消息，输出disjoint。
思路：
首先输入和A题非常类似。
一个点编号，边的数目，（边的数目个）（点编号，对应边权，点编号，对应边权……）
注意到，消息可从任意一人开始传播，也就是说“源点”不是固定的，因此是多源最短路，用Floyd算法，点数最大为100，O(n^3)无压力。
或者用单源最短路算法，然后枚举所有点为源点的情况亦可。
因此存图，跑一遍Floyd，遍历所有可能的最短路即可。
代码中有注释帮助理解。

F
题意：还是最短路问题，不过加了限制——就是题目中的等级观念。即在一条最短路径中，等级最高的人和等级最低的人之间的等级差不能超过M。
思路：
首先注意，这是有向图。
然后输入格式方面，这是中文题，所以应该不难看懂。
这道题难点在于怎么处理等级差限制——

一种方式是枚举等级的范围，设酋长的等级为L，然后等级差最大为M，则枚举等级范围从[L-M,L]，[L-M+1,L+1]，[L-M+2,L+2]……[L,L+M]。每次将不在等级范围内的做标记，使得在跑最短路算法的时候不访问他们即可。

除了枚举等级范围，还可以枚举最小等级、最大等级、等等，实质和做法都是一样

的，这里不赘述。

参考代码给出的是枚举等级范围的方式。

G

题意：

还是最短路问题，但在计算“最短路”的时候多了个条件，就是，当前机器人有一个面向的角度，从一个点到另一个点的时候，如果“朝向”不对，那么就需要转动，设定每转1°消耗1秒。

思路：

首先是建图，将点坐标输入完后，遍历所有点，如果距离小于等于R，则添加边，否则跳过。（无向图）。

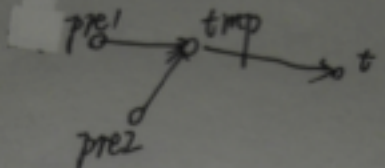
然后是跑最短路的处理。

题目多了个要计算的东西就是“转动的角度”。这里需要用到cmath头文件中的atan2函数，atan2(斜率)返回到倾斜角（弧度制），倾斜角范围为-pi到pi。（科普下pi的写法，通常const double pi=acos(1.0);）

因为要计算角度，那么对于每一个结点i，除了用d数组表示从源点到它的“最短路”以外，还需记录获得这个d值的对应的上一个结点（计算角度用），因此需要一个pre数组。

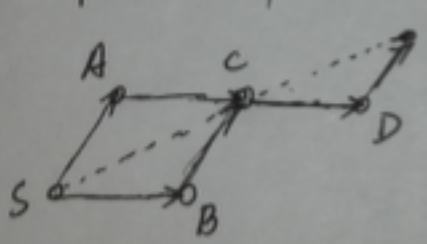
然后每次计算、更新即可。

提一个问题：是否存在一种情况，



从 pre1 与从 pre2 到 tmp 使得得到的 d[tmp] 相等，但从 tmp 到某个结点 t 的时候，虽然是 pre1 更优，但 pre 数组记录的却未必是 pre1 而可能是 pre2，这样的话计算出来的 d[t] 就有可能是错的。

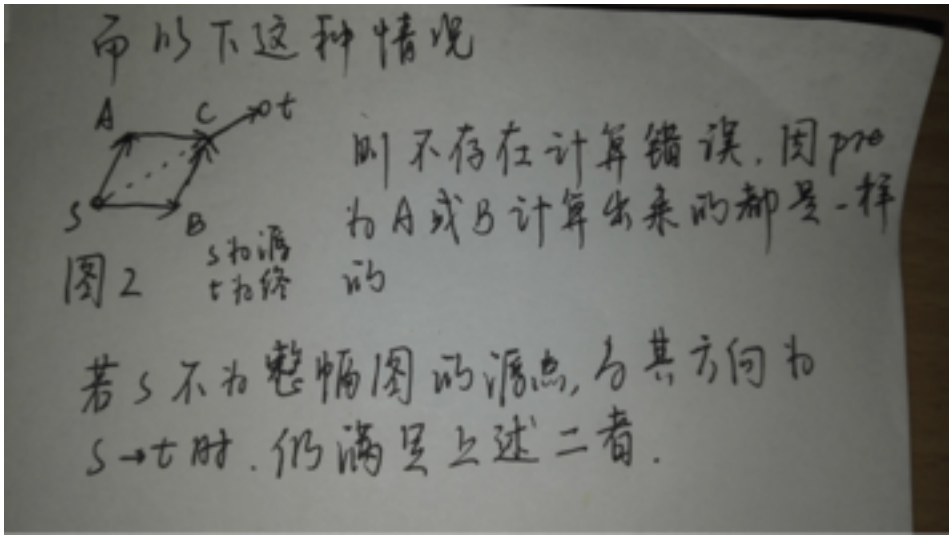
那么这种情况其实是存在的



S 为源点
t 为终点

图 1

ACBS 为菱形，S、C、t 在同一直线上。从此时到达 C 得到 d[C] 的有 pre 为 A 和 pre 为 B 两种情况，虽然 pre 为 A 时才是"最短路"（错误发现在计算 d[D] 时）



事实上，除了d值相等这种情况外，还存在一种状况——
假设对点v进行更新，存在d值较大者与其松弛后的d[v]优于或者说小于d值较小的。可以根据下面的数据自己画图看一下。

解决方法：
这道题难搞在多出了一个“角度”需要计算，正因为这个角度跟它的上一个结点有关，在判断当前是否最优的时候就显得尤为复杂，也导致不能直接用Dijkstra算法搞。
那么我们可以对点进行“拆分”。
对于上一个点是k点的i点的状态，令点编号为k*n+i。
对于边i，j
其权值的难分析性在于还与上一个点有关。
假设上一个点是k。
那么显然这一条边两个端点的编号为k*n+i，i*n+j。
这样的话，对于这n*n个点，就可以在建图后直接跑Dijkstra了，因为这个时候已经没有“角度”捣乱了（角度被直接加进了对应边的边权之中）。

这不是唯一的解决方法。

给两组数据：

```
3 6
0 0
3 0
0 3
3 3
6 3
6 6
```

和

```
3 6
0 0
0 3
3 0
3 3
6 3
6 6
```

考虑上述情况的解法应该得到两个237，否则依次得到327和237。

一组数据：

```
6 6
0 0
5 0
-1 5
5 5
10 5
10 10
```

正确答案269。
但事实上，似乎数据有点水，没考虑到这个情况也能ac。

H
题意：
给出一个迷宫，让依次输出从迷宫入口扶左墙走、扶右墙走、最短路到迷宫出口的距离。
思路：
扶左墙走和扶右墙走用dfs，最短路用bfs。
dfs函数参数除了有坐标x和y以外，还可以有一方向标记，比如说定义面朝上下左右依次为1234之类的，或者用宏定义或者const int up=1之类的，然后在写代码的时候直接用up，down，lef，rig，写代码和调试的时候会方便很多。
大家不妨去看看hhn的代码，很简介，这题我就不献丑了2333。