POJ1125【基础】

题目大意：
有 n 个人，每个人可以把谣言同时传给一些人，传给这些人中的每个人要花费一定的时间。然后这些人又会继续把谣言向自己认识的人传播。但是，A 能传给 B，B 不一定能传给 A。现在要让整个圈子里的人全部知道这个谣言，求最短需要多少时间。

输入格式：
有 K 个测试数据。每个数据的第一行是一个数值 M(M<=100)，表示这个圈子里有多少人。接下来 M 行每一行是第 i 个人的传播渠道，每一行第一个数字 N 表示第 i 个人可以向 N 个人传播谣言。接下来有 N 对数字(Tj,Cj)，分别表示可以向第 Tj 个人传谣，需要的时间是 Cj(Cj<=10)。如果 M=0，表示所有测试数据结束。

输出格式：
对每一个测试数据输出一行，每行两个数 S、T，表示应该交给第 S 个人去开始传谣，谣言传遍圈子需要时间 T。如果谣言无法传遍圈子，则输出 disjoint。

题解：
典型的求多源最短路。每一个人的完全传播用时为该人传播到其他人用时中的最大值。本来应该直接用 Floyd 写的，但是由于只记了 SPFA，我试着用 SPFA 来写，麻烦了一点，但也可以过。

POJ1502【基础】

题目大意：
阿波罗超级计算机有 N 个 CPU，CPU 之间可以通信。一个 CPU 可以同时向所有和它连接的 CPU 发送信息，但是发送到各个 CPU 用时不同。CPU 之间的总线是互通的，也就是如果 CPU A 可以向 CPU B 发送信息，那么 B 也可以向 A 发送信息，所需时间一样。CPU 发送信息到自己不需要时间。现有信息从第一个 CPU 发出，问此信息发送到所有 CPU 需要多少时间。

输入：
第一行一个 N，表示 CPU 数量
第 2 至 N 行，第 i 行有 i-1 个数字或字母 x。第 i 行第 j 个数字表示编号为 i 的 CPU 向编号为 j 的 CPU 发送信息需要的时间。如果是字母 x，表示两个 CPU 之间不能直接通信。

输出：
信息发送到所有 CPU 需要多少时间。

题解：
这题是赤裸裸的邻接矩阵求单源最短路，但是我在上面挂了一个半小时，最后还要找别人的题解来对拍数据。之所以会如此不堪，是因为我写 SPFA 的时候，队列松弛写错成了 while head<=tail do，然后就把循环队列给废了。改了这一点以后马上 AC 了。另外字符串处理现在越来越生疏了，因为很久没有接触了，还要从以前的代码里抄一段回来，落得个不光彩的 16ms AC……

POJ1860【基础】

题目大意：
城市里有些货币兑换点，每个兑换点擅长并只兑换两种特定的货币。不排除有些兑换点兑换相同的货币对。每个兑换点都有自己的汇率。并且定义货币 A 到货币 B 的汇率是 1 单位货币 A 能换到的货币 B 的数量。每个兑换点也要收取一定的佣金，是从源货币中扣除的。例如某个兑换点兑换美元和卢布，兑换率为 29.75，佣金为 0.39，你可以得到（100-0.39）×29.75=2936.3975 卢布。
现在有 N 种不同的货币可以在城市中自由兑换。为简便起见我们对所有货币从 1 到 N 编号。每个兑换点的数据我们用 6 个整数数来描述：整数 A、B 为它可以兑换的货币编号，实数 RAB、CAB、RBA、CBA 表示 A 对 B 的汇率、A 兑换 B 的佣金、B 对 A 的汇率、B 兑换 A 的佣金。你有一定数量的货币 S，希望通过一系列的兑换以后增加你的财富，并且最后要兑换成货币 S。注意兑换过程中你持有的货币数量必须始终为非负。

输入：
第一行包括四个用空格分开的数字，前三个整数第四个实数：N（货币数量），M（兑换点数量），S（你持有的货币种类），V（你持有的货币数量）。
接下来 M 行每一行包含六个空格分开的数字，表示一个兑换点的情况，依次为：A、B、RAB、CAB、RBA、CBA。
对于所有数据，1<=S<=N<=100，1<=M<=100，V 是实数，0<=V<=10^3。对于每个兑换点汇率和佣金都是实数，小数点后最多有两位小数。10^-2<=汇率<=10^2，0<=佣金<=10^2。注意，空格分隔的数字，空格可能是一个或多个。

输出：
如果你能通过一系列的兑换增加你的财产，输出 YES，否则输出 NO。

题解：

这题可以按最短路来算。不过题目里的边没有明确的权，权的形式以兑换比率出现而不是相加。Dist[x]表示兑换成第 x 种货币最多可以有多少数量。则三角形比较方式为 if (Dist[now]-Cost[now,new])*Rate[now,new]>Dist[new] then Dist[new]:=(Dist[now]-Cost[now,new])*Rate[now,new]。
最短路情况下的负权回路，与最长路情况下的正权回路是等价的。因为最短路的判断可以写成 if Dist[new]-(Dist[now]+Weight[now,new])>0 then Dist[new]:=(Dist[now]+Weight[now,new])，而最长路则是把>改成了<，所以原来的负权变成了正权。
SPFA 中负权环的判断是：一共有 N 个点，如果一个点入松弛队列了 N 次，则必有负权环。这个结论是由 Bellman-Ford 算法的正确性推导出来的。
在此题中还有一种比较偷懒的做法是更新过程中 Dist[start]>初始值即退出。这样子只需要 32ms，而数组判断法需要 63ms，将近多了一倍。

POJ1797【基础】

题目大意：
已知有 n 个城市，有 m 条道路连接这些城市，每条道路有一个最大的承载量。一条路径上的最大承载量是该路径各条道路最大承受量中最小的一个。现在要求从城市 1 到城市 n 的最大承载量。假设总能找到一条路从城市 1 通往城市 n。

输入：
第一行是一个整数 K，表示有 K 组数据。
以下每组数组中，第一行包含两个整数 n、m，分别表示城市数 n 和道路数 m。
接下来 m 行每行有三个整数，分别是一条道路的出发城市、目的城市和该道路最大承载量。
其中 1<=n<=1000，每条道路的最大承载量不超过 1000000。

输出：
每一组数据先输出“Scenario #i:”，i 为数据组号。下一行输出一个整数为所求的最大的承载量。然后再跟着有一行空行。

题解：
这题我看了 Discuss 才知道是最大流，但是一样可以用 SPFA 来做。只不过这里的三角形比较的方程需要改变一下，原来是
Dist[new]:=min(Dist[new],Dist[now]+Edges[now,new])，现在改为 MaxLoad[new]:=Max(MaxLoad[new],Min(MaxLoad[now],Load[now,new]))，即以当前点作为中继点，新的点的最大承载量是 max(原最大承载量，min(当前点最

大承载量，当前路径最大承载量))。注意起点到自己的承载量是 maxlongint 需要初始化。这样子的话就可以继续使用 SPFA 了。

这题加上此前查资料（查 1511）让我对 SPFA 有了一种新的理解（或者说我原来的理解就是错误和不全面的），因为这题里面的 SPFA 更像是一种动态规划的状态转移。所以 SPFA 是在有“路径”这一形式的情况下，求从某一点（状态）到另一点（状态）关于某一维度（比如路径，费用，流量等）的最优解都可以用的，关键在于三角形比较方程应该改为和该维度相关的比较方程，而不一定是和路径相关的加减乘除。其实回想一下 SPFA 的诞生，其本身就是和 BFS 高度相似的，不过是加上了边权的计算。而动态规划往往可以写成记忆化搜索，这一搜索的过程便可以按不同的场景由 BFS 或者 DFS 来实现。

另外这题我一开始用的是邻接表（上一题改完 1511），但是第一次提交就 Time Limit Exceeded 了。我观察了一下，估计是释放邻接链表的空间时耗费了太多的时间，于是改了写邻接矩阵，马上就过了，但是内存使用得太多了，比 Pascal 语言通过里面内存最少的要多了 19 倍，比平均水平多了 1 倍。我估计使用前向星的话应该就可以消减一半的内存使用到 4M，但是 400K 是如何做到的还真是想不到啊……

再另外有个很神奇的事情是我交了两次，其中后一次取消了 AddEdge 函数传递的第一个变参（原来脑残了写邻接表的时候为了以后的通用性传了变参，见代码），我本以为会快一点结果却是更慢了。我估计是全局变量和参数变量的内存地址不同，全局变量的寻址时间长一点吧，下次可以找个机会验证一下。

POJ2394【基础】

题目大意：
农夫 John 的农场里有一包谷物从谷仓里被 John 的牛偷走了，John 决定要从他的 C（1<=C<=100）只牛里找到罪魁祸首。幸运的是 M（1<=M<=70000）秒前有一个卫星飞跃了它的农场上空并拍下了当时所有牛的位置，而拍照时盗窃案尚未发生。
John 的农场里有 F（1<=F<=500）块田地依次编号为 1 到 F，被 P（1<=P<=1000）条双向道路连接，这些双向道路的通过时间为 1 到 70000 秒。1 号田是谷仓所在地。在一块田里走路的时间可以忽略不计（假设两条连到同一块田里的路的终点在一起）。
给出 John 农场的照片，请找出所有有可能犯罪的牛。
友情提示：别声明一个叫“time”的变量，否则会调用系统接口让你永远得不到想要的结果。（译者吐槽：尼玛编译器不会报错么系统函数不能赋值啊难道 GCC 那么逆天反正 Free Pascal 守法得很）

输入：
第一行是空格分开的四个整数：F，P，C 和 M
接下来的 P 行每一行有三个空格分隔的整数，用以描述一条路径：起始田地编号，到达田地编号，通过所需秒数。
接下来 C 行依照编号为 1 到 C 的顺序给出每头牛所在田地的编号。

输出：
第一行给出一个整数 N，即有多少头牛可能犯罪。
下面 N 行每行给出一个整数，即可能犯罪的牛的编号。所有编号需要按升序输出。

题解：
单源最短路。以谷仓所在地为源，SPFA 一次到所有各点，然后统计各牛所在位置是否有嫌疑。排序的问题，数据太小，桶排序搞定。

我第一次叫出现了神奇的格式错误，第二次再交过了，POJ 真是个神奇的网站_(:3」∠)_

另外，这道题我看了一下排行数据，瞬间就震惊了。同样是 Pascal 语言，我用了 2928K Memory 和 79MS Runtime，但是排第一的是 36K Memory 和 0 MS！我无法想象是怎么做到的，特别是内存（难道是前向星？回头有空试一下）。然后又想起了李开复的《算法的力量》，所以算法精益求精没有极限，自己也不能那么容易满足啊！

POJ1062【中等】

题解：
题目是中文的，但是理解起来并不那么容易，主要等级问题害死人。其实题目里的表述并不准确，应该是整条交易链上的最大等级差不超过 M。例如 A1，B2，C3，M=1，那么和 A、B 交易了以后就不能和 C 交易了，和 B、C 交易了以后就不能和 A 交易了。我的思路是枚举等级区间，从[level[1]-M,level[1]]到[level[1],level[1]+M]，其中 level[1]表示酋长的等级，然后对每个区间进行一次 SPFA。因为对于所有货物我们可以这样处理：虚构一个点 0 号表示探险家，0 号到所有物品的初始距离和边权均为该物品原价。如果一个物品 A 有替代品 x 和优惠价格 p，那么可以看作是由 x 指向 A 的边，权值为 p。这样问题就变成了从 0 号点到 1 号点的最短路径了，果断 SPFA 之。

另，这题的数据很神奇，我 Wrong Answer 了 10 多次，但是我和网上 ACcept 了的程序进行对拍，把 Discuss 里面的所有数据都对拍了一次都没有异常。但是就是过不了。Discuss 里面也有提到某些数据 ACcept 了的代码未必能过，过了的未必能 ACcept _(:3」∠)_

POJ1511【中等】

题目大意：
P 个志愿者每天早上从 1 号站（总站）出发，前往所有 P 个车站（包括总站）工作（每人去一个站），下午回到总站。各车站之间有单向的行车线路，车辆到达目的地后空驶回起始站。各车站都有所有的行车线路和行车价格让志愿者查阅。求每天所有志愿者乘车费用总和的最小值。

输入：
第一行 N，表示 N 个测试数据。
每个测试数据中第一行为两个整数 P Q（ 1 <= P,Q <= 1000000），表示站数和行车路线数
下面 Q 行每行三个数字，分别表示出发站编号、目的站编号和价格
所有行车价格将小于 1000000000。

输出：
N 行，每行对应每个测试数据的解

题解：
出发好算，以总站为源点一次 SPFA，把到各站点的价格加起来即可。回来的时候我一开始算了 P-1 次 SPFA，效率太低，后来想一下把原图各边反向，再求一次 SPFA 即可。
数据太大，我第一次偷懒用了邻接矩阵（因为此前做的 1125 和 1502 都是邻接矩阵），没有去考虑空间的问题。结果编译器直接出问题了，按 100 万×100 万的规模开数组，我的机子上可以编译但是执行出错，提交了也是 Run Time Error。我只能老老实实用邻接链表写，并且我在每次 SPFA 搜完以后一定会 Dispose 掉那些空间避免爆空间。

这题数据也是相当神奇。原题是 Central European Region of ACM Collegiate Programming Contest 1998 的，我找了原题的数据来拍是不到两秒可以 ACcept 的，但是 POJ 上还是 Wrong Answer 了_(:3」∠)_ 伟大而神奇的 POJ

POJ1724【中等偏上】

题目大意：
有 N 个城市，编号依次为 1 到 N。城市之间有一些道路，每条道路有一定的距离和通过所需的花费。Bob 有一定的钱，现在要求他从城市 1 到城市 N 的最短路径，这一最短路径所花费的金钱应该不大于 Bob 所持有的总金额。

输入：
第一行输入一个 K，0<=K<=10000，为 Bob 所持有的金钱数。
第二行输出一个 N，2<=N<=100，为城市数量。
第三行输入一个 R，1<=R<=10000，为道路数量。
接下来的 R 行每行依次输入四个数 S、D、L、T，分别表示一条路径的出发城市、到达城市、道路长度和花费金钱数，其中 1<=S,D<=N，1<=L,T<=100。

输出：
唯一一行输出从城市 1 到城市 N 的最短路径长度。如果无法到达，输出-1。

题解：
这题是有一维限制条件的最短路，是最短路的一个经典和常见的变形。我一开始上来按传统的 SPFA 写，在三角形判断那里写了三个条件：
(Dist[now].cost+Edges[now,new].cost<K) and
(Dist[now].cost+Edges[now,new].cost<Dist[new].cost) and
(Dist[now].len+Edges[now,new].len<Dist[new].len)，一旦符合则更新 Dist[new].len 和 Dist[new].cost。但是这样出来的结果是错的，我一开始没有想到为什么。改了半天的限制条件都没有改对。
上网查别人的题解，改成了图上动态规划就可以过了。也就是在原有的以距离为维度的三角比较中，加多一层关于费用的循环。更改以后的三角形比较方程是：if Dist[now,j-Edges[Now,i].cost]+Edges[Now,i].len<Dist[new,j] then ……其中 Dist 各下标含义为 Dist[目的点，可用金钱数]=到目的点最短距离。这样，既确保了路径上的最短，也确保了费用上的可行。
返回来说，我第一次写的三个比较条件并列，就会导致某些可用路径虽然更短但是更贵的不被接受。如果取消掉
Dist[now].cost+Edges[now,new].cost<Dist[new].cost 这个限制条件，在上两个条件判断为真并更新新的点的数据时，我又会面临两个分叉：
Dist[new]:=Dist[now].cost+Edges[now,new].cost 或者 if
Dist[now].cost+Edges[now,new].cost<Dist[new].cost then
Dist[new]:=Dist[now].cost+Edges[now,new].cost。如果使用前者，显然不对，因为后面出现某次更新带来了更大的 Cost 值就会破坏解的最优性；如果使

用后者，也不对，因为前两个条件判断为真，则新点的数据必须被更新。所以两个限制条件也是错的。

这一题给我的教训相当深刻啊，因为是第一次接触带限制条件的最短路变形，然后很傻很天真地把所有限制条件一股脑地写到了状态更新的判断那里。这一点以后要注意。

POJ3159【中等偏上】

题目大意：
有一些糖果，班长 flymouse 要奋发给班里所有的小朋友，但是小朋友之间可能会有一些要求需要满足，即某人要比另外一个人多一定数量的糖果。班长 flymouse 想让他的糖果尽量比另外一个同学 snoopy 的多，问最多能多多少。

输入：
输入只包含一组测试数据。第一行包含两个整数 N（<=30 000）和 M（<=150 000）。N 表示小朋友数目（而且小朋友们依次编号从 1 到 N），snoopy 和 flymouse 分别是第 1 号和第 N 号。下面的 M 行每一行有三个整数 A，B 和 c，代表编号为 B 的小孩比编号 A 的小孩最多不能多 c 个糖果。

输出：
一行一个整数，flymouse 可以比 snoopy 多拿的糖果数的最大值。这个值是有限的。

题目提供的提醒：
32bit 长度的整型已经足够满足算法要求了（即 Pascal 的 longint 和 C 的 long）。

题解：
刚刚看这道题我并没有很快想到如何建图。从常规的思路来说，给出的每一组（A，B，c）肯定可以建一条边，并且 c 应该是权。进一步地，既然 c 是权，那么 Dist[k]的含义，就应该是第 k 号相对于某一个人最多可多持有的糖果数。联想到 POJ1062，我差点想建一个虚拟起点来给 Dist[k]作为参考对照。但是后来转念一想，既然是“相对”的，而且题目又要求第 n 个人比第 1 个人可以多的糖果数，那么干脆全部转为以第一个人作为参照。这样，只需要满足 Dist[new]-Dist[now]<=Weight[now,new]即可。结合最短路中的三角形不等式，可以写成 if Dist[new]>Dist[now]+Weight[now,new] then Dist[new]:=Dist[now]+Weight[now,new]。注意，在建图的时候是有向图！

如果这道题仅仅是如此，那么就不是【中等偏上】而仅仅是【中等】了。因为，按照常规写邻接链表+队列的 SPFA，是会挂掉的。我第一次也是这么挂了，然后去查 Discuss 和别人的题解，大家纷纷表示这样肯定会挂，要用栈和数组存图。栈代替队列的话好写，而且因为 SPFA 的松弛顺序是不要紧的，不像广搜需要分深度逐层搜索。因为图可能存在负环，而栈式的 SPFA 就是深搜式的，如果存在负环，会比广搜式的更早发现，因此更节省时间。第二，数组存图，是因为链表的速度太慢了，每次 new 完再赋值自然不如程序启动的时候就申请完所有的内存来得快。但是链表比起下标代表起始点的边表和邻接矩阵，乃至前向星，最大的好处就是内存节省，需要多少申请多少，边表和邻接矩阵基本上会浪费掉大部分的内存。前向星在速度和内存使用上比较平均，既有数组的直接，又刚刚可以满足最大数据范围需求的存储，但是需要对起点进行排序，如果输入不能满足起点的降序或者升序的话就基本上不能用了。所以引入一种新的数据结构：数组模拟链表。具体的实现看代码和注释。

最后是 496ms 和 3616K 内存过掉了。看 Status List 看得我心惊肉跳，有人用 G++（GNU C++）47ms 刷掉了，内存只有 2180K！完全无法想象是什么算法……倒是用 Pascal 写的基本上都是 300 多到 400 多 ms 和 3000 多 K 内存，估计写法都大同小异吧。

上网查题解的时候才知道这题其实是差分约束系统，但是可以用最短路做。至于是为什么，我现在还没学差分约束系统，留待以后再解决这个问题吧。

其他的一些最短路的题目，我看了没有做，但是给出译题和简单的思路。

POJ2240【易】

题目大意：
套利是指利用不同货币之间的兑换率的差来将原来的某一种货币通过兑换活得更多该种单位的货币。现在需要你写一个程序来判断根据给定的兑换率能否进行套利。

输入：

输入会包含一组或多组测试数据。每组测试数据的第一行是一个整数 N（1<=N<=30），代表有多少种货币。接下来 N 行每一行包含有一种货币名，并且不会出现空格（比如 USDollar 和 BritishPound）。再下一行包含一个整数 M，代表有多少种兑换方式。接下来 M 行每一行表示一种兑换方式，格式为“原货币 兑换率 新货币”，三个参数中间用空格分隔，兑换率是一个实数。在列表中没有出现的兑换方式是严格禁止的（即 A 可兑换 B 不代表 B 可兑换 A）。每个测试数据组之间有一个空行进行分割。N=0 时，输入结束。

输出：
对于每一个测试数据组，输出一行“Case %i:%ans”，%i 代表测试数据组编号，%ans 为“Yes”或者“No”表示能否进行套利。

题解：
和 POJ1860 基本一样，还要更简单一点。SPFA 过之。

POJ2253【易】

题目大意：
青蛙 Freddy 在湖中间的石头上看到它的女神 Fiona 在另一块石头上。Freddy 想和女神呆在一起，但是水像恒河水一样它不敢下去，只能跳到水面的石头上。女神的石头似乎很不幸不在 Freddy 一跳登天的范围内，所以 Freddy 需要跳到其他石头上作为中转。Freddy 想知道它跳到女神所在的石头上的最短路径中最大的一跳需要跳多远。

输入：
输入中包含一组或以上的测试数据。每一组测试数据的第一行是一个整数 n（2<=n<=200）。接下来 n 行，每一行包括两个整数 xi,yi(0<=xi,yi<=1000)代表第 i 块石头的坐标。Freddy 在第 1 块石头上，Fiona 女神在第 2 块石头上，剩下的 n-2 块石头上没有别的备胎。每组测试数据中有一个空行分隔。N=0 时，输入结束。

输出：
对每一组测试数据，输出一行“Scenario #x”，#x 代表数据组编号（从第一组数据开始输出），再输出一行“Frog Distance = #y”，#y 代表最大单次跳跃距离，保留小数点后三位。两组输出之间需要有一个空行进行分割。

题解：

和 POJ1797 基本一样，不过是把路径最大载重变成了最小跳跃距离，把路径中最小的最大承载量变成了路径中最大的最小跳跃距离，就是改个三角形判断的大于小于号罢了。SPFA 过之。

POJ2387【易】

题目大意：
奶牛 Bessie 在田野里想尽快回去谷仓睡觉。它所在的 John 农夫的农场有 N(2<=N<=1000)个地标，分别编号为 1..N。地标 1 号在谷仓处，Bessie 所在的苹果树下有地标 N 号。奶牛们在农场中移动都在地标之间 T（1<=T<=2000）条双向牛行道上走，特别是路痴 Bessie 只会在牛行道上走而不会走出去。现在已知各地标和地标间的牛行道长度，求 Bessie 所走的最短路径的长度。假设 Bessie 总能找到路走回谷仓。

输入：
第一行有两个整数：T（牛行道数）和 N（地标数）
接来下 N 行每一行有三个用空格分隔的整数，前两个是某条牛行道两端的地标编号，第三个是该牛行道的长度（长度不大于 100）。

输出：
一行，给出从地标 N 号到地标 1 号的最短路径长度。

题解：
注意是无向图，可能有重边，即两个点之间有两条或更多的边，取最小的一个即可。剩下的就是赤裸裸的最短路，SPFA 过之即可（按说 SPFA 可以无视掉重边的，只是慢一点而已）。另外 Discuss 里提到有数据范围错误，还有 Discuss 里有很多人抱怨写惯了最短路都是先给点数再给边数现在反过来害死了一堆人，我只能呵呵了这种老油条的风气绝对要不得，不过如果不是自己现在逼着自己为了适应英文环境而每题必仔细看原文，可能自己也会掉坑里去的。

POJ3259【易】

题目大意：
农夫 John 在给自己的农场进行土地勘察时发现了一系列令人吃惊的虫洞！每个虫洞都是非常独特的因为它可以把你单向传递到它的出口，并且到达出口的时间会早于你进入虫洞的时间！John 的农场包括 N（1<=N<=500）块田地，依次编号为 1 到 N，并有 M（1<=M<=2500）条道路，和 W（1<=W<=200）个虫洞。John

痴迷于时间旅行，所以他想试着从某一块田地出发，通过穿越某些虫洞和走过一些道路，然后回到出发点并且早于他出发的时间。
你将拿到 John 所有的 F（1<=F<=5）个农场的地图。在他的所有农场中，没有哪条路需要走超过 10000 秒，也没有哪个虫洞可以让时间倒退多于 10000 秒.

输入：

第一行 F。下面依次给出 F 个农场的数据。
每个农场数据的第一行有三个空格分隔的整数，依次是 N，M，W。
接下来的 M 行每行有三个空格分隔的整数 S，E，T，表示编号为 S 和 E 的田地中有一条双向路径，用时为 T 秒。注意，两块田地中可能有不止一条路径。
接下来的 W 行每行有三个空格分隔的整数 S，E，T，表示有一个虫洞可以把人从编号 S 的田地送到编号 E 的田地，并让时间倒流 T 秒。

输出：
F 行。对第 1 到第 F 个农场，每一行输出一个“YES”或者“NO”，代表这个农场能否让 John 穿越时空见到以前的自己。

题解：
求负权回路，可以用 SPFA。注意虫洞是单向的，其他各点之间是双向的。

POJ2502【易】

题目大意：
你搬到了一个大城市，每天要到很远的地方上学，所以你决定坐地铁和走路结合而不是像以前在村子里一样骑自行车，并且你希望知道最少需要多少时间从家到学校以免迟到。你的步行速度是 10km/h，地铁的速度是 40km/h，并且你有春哥保佑，每次到地铁站必定马上有地铁可以开出。你不需要考虑坐地铁的花费，并且你可以进行转乘。

输入：
第一行包括四个空格分隔的整数：Hx,Hy,Sx,Sy，分别代表你家的横纵坐标和学校的横纵坐标
接下来有不确定的行数，每一行都给出一条地铁线路的站点。每行有若干个正整数，两个两个依次表示这条地铁线路上每一个站的横纵坐标。你可以假设站点间的轨道是直的。每条线路最少会有两个站点。当两个坐标都是-1 时，表示这条地铁线路结束。

输出：

一行，你从家到学校最少所需的分钟数，四舍五入。

题解：
此题关键在于建图。对于所有的点，先两两给一个权 weight:=distance/10，然后对铁路上的相邻两点更新其权为 weight:=distance/40。然后就是裸体的最短路，SPFA 过之。

POJ3615【中等】

题目大意：

一堆奶牛要练习跳高，但是它们怕累，所以想尽量节省体能。它们不担心跳过一些小的障碍物，只关心需要跳过的最高的障碍物的高度。

奶牛们的练习室里有 N(1<=N<=300)个停留点，依次编号为 1 到 N。有 M(1<=M<=25000)条单向路径从 Si 通向 Ei，每一条路径中有一个高为 Hi(1<=Hi<=1 000 000)的障碍物。

奶牛们有 T(1<=T<=40000)个训练任务需要完成。第 i 个任务包含两个数字 Ai、Bi，表示要从第 Ai 号停留点跨越障碍物到 Bi 号停留点。你不需要考虑奶牛们如何到 Ai 和从 Bi 回来。奶牛们希望在每个任务中跳过的最高障碍物高度最小。

输入：

第一行是三个用空格分隔的整数 N，M 和 T

接下来 M 行每行依次有三个整数 Si，Ei 和 Hi。

接下来 T 行每行一次有两个整数 Ai 和 Bi。

输出：

T 行，每行输入一个整数，表示从 Ai 到 Bi 的最小的最高障碍物高度。

如果无法到达，输出-1。

题解：

和 POJ2253 的原理一样，只不过是要做多组数据的计算。

POJ3268【中等】

题目大意：
有 N 个农场（1<=N<=1000）依次编号为 1..N，每个农场有一只奶牛代表该农场去农场 X（1<=X<=N）参加奶牛大会。一共有 M（1<=M<=100000）条单向的牛行道连接各个农场，每条牛行道耗费的时间都不会超过 100 个单位时间。
奶牛们都很懒并且很聪明，一定会选最短的路径从他们所在的农场到聚会农场再回到他们所在的农场。由于牛行道是单向的，所以它们有些牛可能来回走的路是不一样的。牛行道足够宽可以容纳足够多的牛同时通过。现在要求走去聚会地点再走回来所需要的时间最长的那头奶牛所需要的时间。

输入：
第一行是三个空格分开是整数，依次是 N，M 和 X
下面 M 行每一行有三个空格分开的整数，依次是每一条牛行道的出发农场、到达农场和所需要的单位时间。

输出：
一行，一个整数，代表走去聚会地点再走回来所需要的时间最长的那头奶牛所需要的时间。

题解：
和 POJ1125、POJ1511 有点类似，先正反向求一次最短路，然后求两次最短路中到各点距离的最大值。SPFA 过之。