CONTEST	JIANG, Yanyan's Contest #2
Location	Zju Online Judge 2277 ~ 2283
URLs	Http://acm.zju.edu.cn/show_problem.php?pid=22xx
Author	LoveShsean (Wu Chenyang)

PROBLEM SOLUTION

The Gate to Freedom

方法： 数学
复杂度： O(1)
讲解：作为第一题，当然不会太难。只是简单的数学计算，求 N^N 的第一位数字。我们首先要注意一个事实，10 ^ x 这个数，如果要求它的第一位数，那么 x 的整数部分是完全没有用的，即 First Figure (10 ^ x) = First Figure (10 ^ {x})。那么可以这样做，N ^ N = 10 ^ (lg (N ^ N)) = 10 ^ (N*lgN)，只要求出 NlgN 的小数部分 K，取 10 ^ K 的整数部分就是答案了。

Fight for Food

方法： DP + 优化
复杂度： -
讲解： 首先枚举起点，对整个地图进行 BFS，求出每对点之间的最短距离。然后我们很容易得到一个 O(P^2)的类似于 LIS 的算法，但这样铁定超时，我们需要优化。考虑我们首先做得把所有老鼠的出现时间按大小排序，那么这个限制了我们转移的范围，如果当前求得是 f [i]，那么如果存在一个 j > 0，使得 t [i] - t [j] > 离(xi, yi)的最远距离（预处理求得），那么显然在枚举 k < j 来转移是没有意义的。这样优化之后，效果非常好。很快地 Accept 了。

In the Mirror

方法： BFS
复杂度： O(NM*2^P)
讲解：题目看起来很复杂，对于这种题目，我们要找的只是一种可以接受的状态表示方法。所谓“可以接受”，就是实践上和空间上都能承受的意思。这个题中镜子的数量不多于 10 个，这个是本题的突破点。为什么镜子只给了 10 个，而且每个镜子的状态只有两种，‘/’ ‘\’，这提示我们用状态压缩的方法，用一个二进制数表示所有镜子的状态，最多共 2^10 = 1024 种状态。不妨以 F [S] [x] [y]为状态，表示镜子的状态为 S，我到达(x, y)点需要的最少时间，由于每走一步和翻转一次镜子都需要 1 个时间，显然可以舍弃 DP 而用 BFS 来提高效率，这样把 F 数组改为 Boolean 型，访问过的不再访问。实现起来细心一些，还是很容易 Accept 的。

Key to Freedom

方法： DP
复杂度： O(N^3)
讲解：本题唯一特殊的地方就是一棵树，而且数据范围不大，提示我们用 Tree DP。和其他的 Tree DP 相似，本题不是二*树，首先要将其转为二*树，然后以 F [i] [j]为状态，表示在 i 和 i 的所有右兄弟这些树中，当前处在 i，拥有 j 个能量时所能拿到的最大宝物值。显然满足无后效性和最优子结构。那么转移就是

```
F[i][j] = Max {
      F[LeftSon[i]][j' ] + F[RightSib[i]][j-j' ] + Key[i]
                (j >= cost[i] && 0 <= j'  <= j),
      F[RightSib[i]][j] }
```

这样答案就是 F [1] [Power]。总的时间复杂度是 O (n^2 * n) = O(N^3)。

Way to Freedom

方法: Method1: DP + Heap Method2: Greedy + Disjoint
复杂度: Method1: O(NlogN + M) Method2: O(Mlog*N + M)
讲解:
Method1:无向带权图，不仅让我们想起 Dijkstra 求最短路，然而这个题不是求最短路，不是求一些路径的和，而是求一些路径的最大值。那么我们把方程改一下就可以了。
Opt [i] = Max { Min (Opt [j], W (j, i))} (Edge j -> i exists)
这个方程依然满足 Dijkstra 的性质，用堆来实现，复杂度为 NlogN。
Method2:我们考虑如下的贪心算法：先将边按边权从小到大的顺序排序，对于每一个点建立一个集合，将边扫描一便，一条一条地连，连一条边就合并 i 和 j 所在的集合。当 s 和 t 合并到一个集合里时，最后连的一条边的权就是答案，复杂度为 O(MlogM+Mlog*N)。如果用堆或快排-划分来代替排序，复杂度降到 O(M+Mlog*N)。

Challenge of Bravery

方法: 近似算法
复杂度: -
讲解: 我的搜索很弱，只好用一些“无耻”的方法。这个题就是原汁原味的 TSP 问题，我们用模拟退火算法。这里简单介绍一下:
重复多次下面的操作:
1、随机生成一条路径。
2、对此路径进行调整，每次找到 i<j，使得 1->2->3-> ... ->i-1->i-> ... ->j->j+1-> ... ->n 的总长度大于 1->2->3-> ... ->i-1->j->j-1-> ... ->i->j+1->...->n 的长度，那么将 i 到 j 的这个部分路径反过来，完成一次调整。直到不存在 i,j，完成对此路径的调整。
3、用总长度更新 Ans。
这样将会逐渐逼近最优解。不过这个方法不是精确算法，我随机到 10000 次才过掉。呼唤强人讲一个搜索的算法……

Challenge of Wisdom

方法: LIS
复杂度: O(NlogN)
讲解: 为了讲这个题，我们先证明一个定理，为了方便本题，把定理的证明叙述成和本题一致的形式。
定理: 最长上升序列的长度等于不上升序列的最小分划(即将平面上的点分划成

尽可能少的不相交的不上升序列)。

证明: 一个显然的结论是最长上升序列的长度小于等于不上升序列的最小分划。因为上升序列中任意两点都不可能属于同一个不上升序列。也就是说, 最长上升序列中的所有点分属不同的不上升序列。所以, 不上升序列的分划数最少也不会少于最长上升序列的长度。关键的是要证明, 最长上升序列的长度大于等于不上升序列的最小分划。我们来构造一个不上升序列的分划。首先在二维 Euclid 空间中取出所有的满足如下性质的点 (x,y): 对于任意的点 (x′,y′), 总满足 x′<=x || y′<=y,即 (x,y) 的右上方没有别的点。可以证明,取出的点集 {(x,y)} 是一条不上升序列。因为点集中任意两点 (x1,y1) 和 (x2,y2) 总满足 x1<=x2 || y1<=y2 且 x2<=x1 || y2<=y1, 整理一下即得 (x1<=x2) xor (y1<=y2) == true。所以, 把这些点按 x 升序排列后, 得到的 y 相应的成降序。将上面取出的点从空间中去除后, 重复上述过程, 又可以得到一条新的不上升序列。如此反复可以得到一个不上升序列的分划(现在还不能肯定这就是最小的分划)。由前面的结论知道, 这个分划数必定大于等于最长上升序列的长度。我们对得到的不上升序列分划进行分级, 先取出的等级最高, 最后取出的等级最低。这样就得到 k 条分属 level 1、level 2 ... level k 的不上升序列。这些链上的点满足这样的性质: 对于一个属于 level i (i < k) 链上的点 (x,y), 必然存在一个属于 level i+1 的点 (x′,y′), 使得 x <= x′; y <= y′。否则 (x,y) 在取 level i+1 链时就会被取走, 不应属于 level i。从 level 1 上的一点 (x1,y1) 开始, 取 level 2 的点 (x2,y2), x2>=x1 ,y2>=y1, 然后取 level 3 的点 (x3,y3) ..... 最后必然能取到 level k 上的点 (xk,yk)。如此得到的序列 (x1,y1) (x2,y2) ... (xk,yk) 就是一条上升序列。所以, 我们前面得到的不上升序列的分划数就不可能大于最长上升序列长度。这就证明了最长上升序列的长度大于等于不上升序列的最小分划。再加上最长上升序列的长度小于等于不上升序列的最小分划的结论,就证明了最长上升序列的长度等于不上升序列的最小分划。

从证明中, 我们就得到了本题的算法: 先按 x 坐标从小到大排序, x 坐标相等的按 y 坐标从小到大排序。然后拿出排序后的 y 坐标, 求最长下降子序列。其长度就是本题的答案。