**题解**

第一题 高楼

因为高楼高度两两不同，不妨从高到低来排列。设为放了最高的n个楼，且左边和右边分别能够看到l和r个楼的方案数，不难得出：

这个算法是的。

不妨只考虑一边的情况。设为放n个楼房，左边看到k个的方案数。该状态递推同样简单：

回到原问题，我们盯住最高的楼房，枚举它左边的楼房个数，则问题的答案为：

C为组合数。

该算法的时间复杂度为。

第二题 朋友

朋友这道题是一个明显的最短路模型题。

【算法1】

直接建图。

以矩阵的每个格子为点，

如果从u可以弹到v那么建一条从u到v，长度为u点弹射费用的边。

然后从X，Y，Z三个点分别开始做单源最短路。

此时图的尺寸最坏是点数，边数。

具体实现：

Bellman-Ford算法，时间，空间，期望得分20。

Dijkstra算法，时间，空间，期望得分20。

SPFA算法，时间，空间，期望得分20。

以上3个算法的瓶颈在于空间。事实上，对于40%的数据，我们发现图并不是很稠密，可以用邻接表等数据结构来存储连通信息。这样Dijkstra算法和SPFA算法可以得到50分。使用优先队列（堆）进行优化以后可以多得到少许的分数。

注意，在这个图上点数和边数同级，所以当模型是边数很多的稠密图时，堆优化的效果就会显得不明显。

（设V是点数，E是边数。堆可以把的最短路算法优化成，当E越小时，堆的效果越突出。）

【优化1】

思考Dijkstra算法的特点。事实上在这题中我们只要求从一个点到另外2个点的最短路，也就是说，一旦另外两个点的最短距离已经算出，我们就可以终止Dijkstra算法避免浪费时间。原因是，Dijkstra算法运行时，每次确定的点都一定是最短的。

加上这个优化以后会少许提高分数。

注意SPFA算法无法进行此项优化。因为SPFA和Bellman-Ford同属迭代型的最短路算法，算法结束前我们无法保证任意一点的答案是否不会改变。

【优化2】

不使用邻接表和其他任何数据结构来建图，直接根据图的特点来做最短路。

这是一个很节约空间的方法，事实证明内存空间访问次数减少以后，算法的时间也会有很大改善。

进行此项优化可以少许提高分数。

【算法2】

我们发现从最短路算法上的优化来考虑没法得出效率更高的解。所以我们研究图的特点。

在此，出题者发现一种图的改建方法，可以减少边数：

取图中最远可能的弹射距离S。（S≤N+M-2）

引入“云端”这个新的概念。

我们建立一个包含N\*M\*(S+1)个点的图 G[1..N][1..M][0..S]

其中G[x][y][0]代表地面，也就是街区。

G[x][y][h] (h>0) 代表一个云端节点。

当居民在云端节点上时，他必须选择往低一层的相邻云端降落。

也就是说，G[x][y][h]有5条出边：

G[x][y][h]到G[x][y][h-1]

G[x][y][h]到G[x-1][y][h-1]

G[x][y][h]到G[x+1][y][h-1]

G[x][y][h]到G[x][y-1][h-1]

G[x][y][h]到G[x][y+1][h-1]

它们的边权都是0.

也就是说，在高度为1的云端可以落到地面的5个相邻点。

在高度为h的云端，最终一定会落到地面距离不超过h的某个点。

弹射装置的作用，实际上就是把飞飞侠从地面弹到同经纬的一定高度的云端上。

如果A[x][y] = v（弹射能力），那么我们建边G[x][y][0]到G[x][y][v]，边权(长度)为B[x][y]（弹射费用）。

这样一来，每个云端点只有5条出边，每个地面点只有1条出边。也就是说，这是一个相当稀疏的图，点数和边数同级，我们建立了一个点数和边数都是O(N\*M\*S)的图，同样是求3个地面点互相的最短路。

我们发现新图比原来的图显得更自然，注意S与N和M同级。所以，虽然点数增加了，但这个图的边数是算法1中图的约1/N。

于是，堆优化就可以很有效的优化算法了。

堆+Dijkstra算法加上前面说到的优化1和优化2，就可以获得本题的满分了。

最坏时间复杂度：。空间复杂度。

第三题 纸带

这道题主要是考察离散化和数据结构。

【算法一】

建立一个数组，表示每一个格子上面涂写的数字，每次操作直接修改。

时间复杂度，空间复杂度。

期望得分：20分。

【算法二】

用一个队列维护当前保留的线段集合。每当一个新线段加入目前的集合时，检查集合内的所有线段，如果分割某线段则分割线段并答案加一，覆盖某线段则覆盖该线段，如果和某个

时间复杂度，空间复杂度。

【算法三】（非必要算法）

算法一的线段树优化办法，需要进行离散化。

时间复杂度，空间复杂度。

【算法四】

从正面来想似乎没有什么思路，那么不妨从反面思考，从后向前思考这个问题。

如果说从后向前考虑每条线段，有一个优势是如果一条线段被涂过了，那么之后就再也不用考虑这些被涂过的线段。那么需要寻找一个数据结构来满足我们“跳”的需要。

并查集就能够满足我们的需求。

设在某个时刻，为包括i点在内的，在i之前第一个空白点。

那么一条线段没有被之后（原题意义的之后）覆盖，当且仅当。那么判断可以在的时间完成。

那么修改操作呢？如果一个点P被新涂色了，那么可以使。

F数组是显然可以通过并查集优化的，那么这个问题就解决了。

时间复杂度（忽略排序时间复杂度和并查集的常数），空间复杂度。