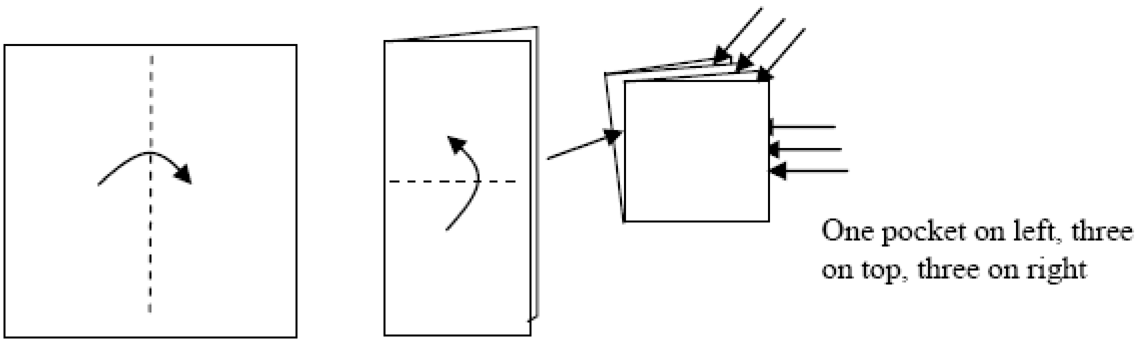
《Pockets》解题报告

南京外国语学校 王悦同

【题目描述】

日式的折纸手工——一种折纸的艺术——常常需要用到“口袋”的概念，特别是在那些复杂的、各个折纸模型需要通过口袋来实现嵌套的作品中。在这题中，我们需要数一数，一个正方形纸片折叠后形成了多少个口袋。一个口袋定义为：在折好的纸片中，从边界上可以看到的一个开口（两层纸之间的开口）。注意：一个开口可能被记作多个口袋，因为它可能可以从多个方向看到。图 1 展示了这样一个例子。注意到，“中间”的那个开口（第二和第三层之间的开口）被算了三次。

图 1：



我们假设这个纸片一开始平放在桌面上，并且任意时刻都不会被完全从桌面上拿起。每次折叠要么是水平的、要么是垂直的，并且折叠只能沿着预先标好的 N 条水平线和 N 条垂直线。没有折叠的时候，水平线从上到下 1~N 标号，垂直线从左到右 1~N 标号。每次折叠，我们沿着一条线，和一个方向，把当前的矩形折成一个更小的矩形。例如，'2 U'表示沿着第二条水平线向上折；'1 L'表示沿着第一条垂直线向左折。折叠后，一条线可能有多个表示方法。（例如，图 2 中，第一次折叠后，'1 D'和'3 D'是等价的指令了）。最后，整个纸片被折成 1*1 的正方形，这时我们来统计它的口袋数量。保证每次操作都是合法的，并且这里我们忽略纸片的厚度。

图 2：

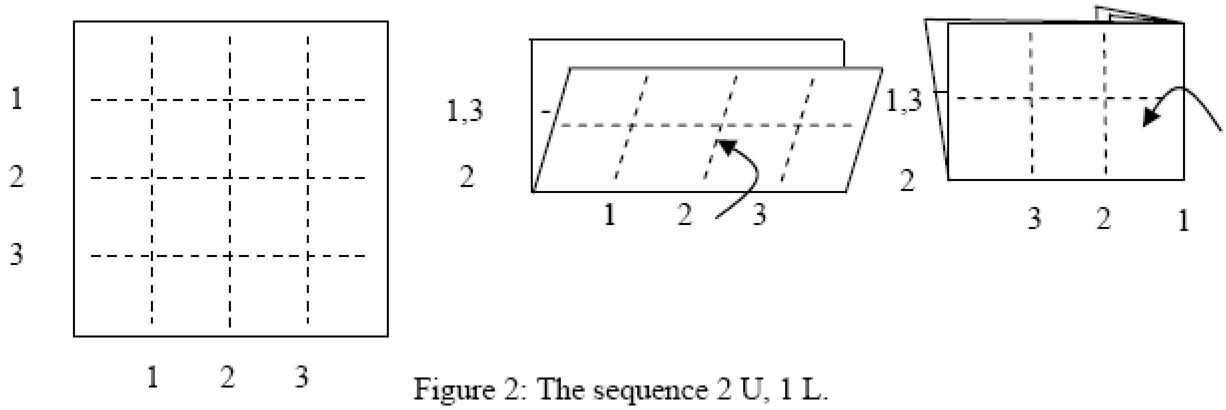


Figure 2: The sequence 2 U, 1 L.

纸的边长、折纸次数<=64，保证最后折成了 1*1 的正方形。

【题目大意】

给一张纸，每次可以按某个方向沿某条线折叠（线水平或垂直，且两两间距相等（这一点原题好像没说？不过应该显然）），最后保证折成了 1*1 的正方形。问这个 1*1 的（厚)纸片每个边上有多少开口。开口即题目中的 pockets，定义在上述题目中给出。

【算法分析】

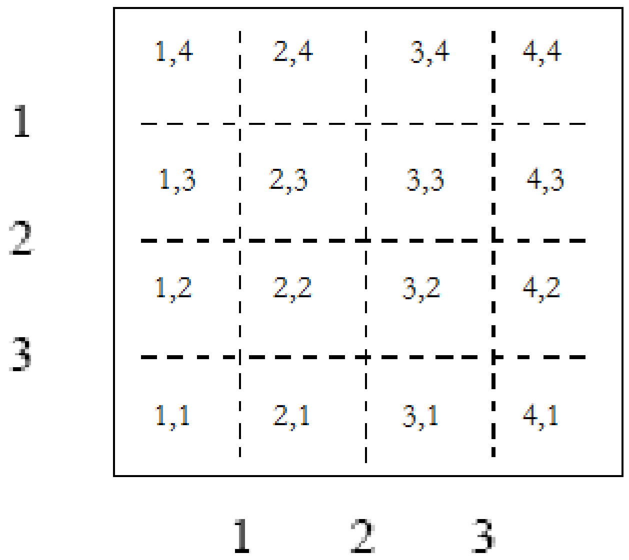
这道题如果论做法，那么比较显然——因为这是一道模拟题。那么模拟题的关键就是如何实现。这题要模拟的是一个折纸的过程——是一个三维的过程，那么我们需要合适的结构来存储、需要合适的方式来模拟折叠的过程。

下面的分析中，由于本题规模很小（64），故所有提到的操作都是为了实现模拟，而不考虑优化的问题。直接用最暴力的做法模拟过程，就可以 AC 了。

首先我们考虑如何存储这个纸片的形态。由于纸片的形态可以千奇百怪——只保证俯视图是矩形，而中间可能每层高度都不一样、中间可能有空心等。因此，我们直接记录最初的 N*N 个格子每个现在在哪儿。用一个数组 locate[i][j]存放三个域，即可表示最初的格子(I,j)现在所处位置的三维坐标。这里三维坐标只是相对坐标——可能是负数，并不影响处理。

我们考虑模拟折叠的过程。折叠时，我们输入一条折痕和一个方向。不过不难发现，输入的折痕现在在哪里我们并不知道！因此，我们还需要一个数组进行维护：vert[i]表示最早的第 i 条垂直折痕现在在哪儿（在哪儿定义为：它的 x 坐标是多少，也就是沿着他折叠相当于是沿着哪一个 x 坐标折叠），还有 horz[i]表示水平折痕。

每次折叠的时候，我们沿着折痕，处理某些点三维坐标的变化。以‘L’操作为例：



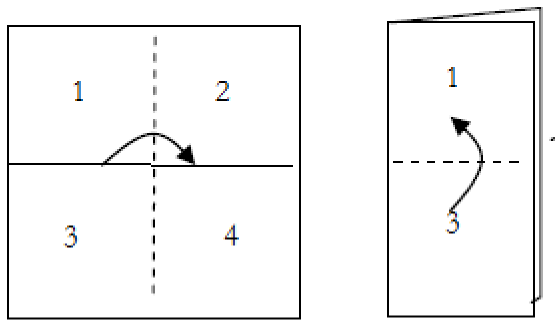
初始时 x、y 坐标如上所述，z 都是 0.

假设沿着 x=t 进行 L 操作，那么对于一个点：x_new=2*t+1-x,y_new=y。z_new 比较麻烦，不过考虑到是相对高度，所以我们假设折叠中，动的一半纸片最高高度是 H1,没动的最高高度是 H2，那么 z_new=H2+(H1+1-z)。这样可以保证相对高度的正确性——尽管中间会空一些，但无所谓，这里要的都是“相对”的。注意，高度理论上是指数级增长的，因此尽管操作数是 64 次，我们仍然需要 long long 存储（实际上这样的数据极难构造；tsinsen 上的数据没有这样的情况）。

沿着 x=t 进行操作后，某些 vert[i]也要改变：如果 vert[i]>t,那么 vert[i]=2*t-vert[i]

因此这样的话折叠操作是可以维护的。

下面考虑统计答案：统计答案时，所有格子的 x,y 都相同，唯一有用的是 z 坐标决定了它们从上到下的顺序。那么什么和开口有关呢？显然是原纸片中的那些相邻格子间的边。不过这个时候遇到了一个麻烦事，考虑如下情况：

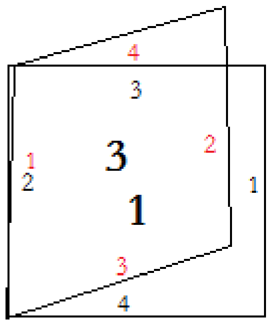


1 和 2 之间，3 和 4 之间，一边有开口、一边没有。也就是说，如果只是根据格子在最初纸片中的标号，我们不能确定哪些地方有边哪些没有。因此我们需要维护一个方向：、

对于每个初始格子，维护 direct[i][j]=(North,South,West,East)，表示这个格子目前朝北、南、西、东的分别是哪个数字。不妨设一开始分别为 1、2、3、4，同时认为东南西北的方向不会改变，那么每次折叠的时候，对应朝向的数字必定会有所改变。

最后根据这些朝向和数字，就可以维护哪里有边。例如，一开始上图中 1 号格子的 2（南）和 3 号格子的 1（北）有边，那么最后如果在某个朝向上，1 号格子是 2、3 号格子是 1，那

么这里有一条边；反之，如果 1 号格子是 1,3 号格子是 2，就没有边了。



上图中，粗体数字表示格子编号（对应之前的图中 1、3 号格子），小黑数字表示 1 号格子上的标号、小红数字表示 3 号格子上的标号。从上图中可以看出，1 和 3 之间只有一条边，并且可以由数字的对应关系确定。（东西南北只是一个固定不变的参考系，最后统计答案的时候并不需要）。

最后 1*1 纸片四条边上的“有无边”的信息存好后答案就容易得到了，直接扫描统计即可。每个格子四个朝向上的数字显然可以由折叠的时候顺带维护；因此本题就解决了。根据实现方法不同，本题时间复杂度在 O(N^2*K)到 O(N^3*K)之间，不过显然都足以通过本题。