Dancing Links

舞蹈链

问题 1：

集合 s 有 n 个成员，现在有 m 个子集，每个子集包含一些成员，每个成员都属于集合 s。在 m 个子集中选择一些子集组成子集的集合 t，使 t 中包含的成员可以覆盖集合 s，即 s 中所有成员都属于 t 中的某个子集。

重复覆盖：集合 s 中的任意成员 x 至少属于 t 中的一个子集，允许同时属于两个及以上的子集。例如集合$s = \{0, 1, 2, 3\}$，在子集$a = \{0, 1\}$、$b = \{1, 2\}$、$c = \{1, 3\}$中选择$t = \{a, b, c\}$即可重复覆盖 s。

精确覆盖：集合 s 中的任意成员 x 属于且只属于 t 中的一个子集，不能出现 x 不属于 t 中的任何子集，或者 x 同时属于 t 中两个及以上的子集。例如集合$s = \{0, 1, 2, 3\}$，在子集$a = \{0, 1\}$、$b = \{1, 2\}$、$c = \{2, 3\}$中选择$t = \{a, b\}$即可精确覆盖 s。

给定集合 s 和 m 个子集，求出其重复覆盖和精确覆盖。

重复覆盖解法：

遍历集合 s 中每个成员 x，若其尚未被包含在 t 中，则在 m 个集合中寻找一个包含 x 的子集加入 t 中，重复该步骤即可获得重复覆盖。

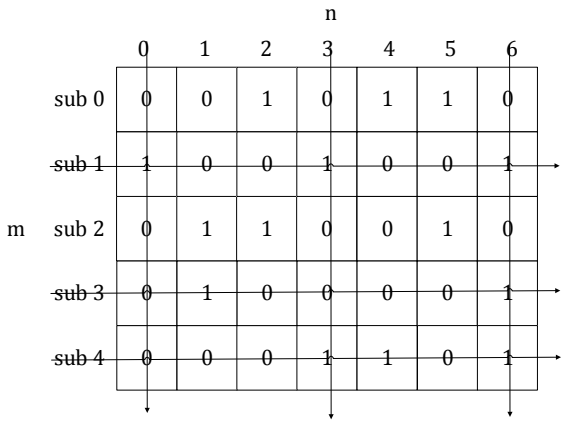
精确覆盖解法：

求解精确覆盖的算法称为 X 算法，将集合 s 中 n 个成员看作列，将 m 个子集看作行，组成一个$m \times n$的矩阵 d。若子集sub_i（其中$0 \le i < m$）包含某个成员x_j（其中$0 \le j < n$），则$d[i, j] = 1$；若不包含则$d[i, j] = 0$。对于集合$s = \{0, 1, 2, 3, 4, 5, 6\}$有$n = 7$个成员，还有$m = 6$个子集$sub_0 = \{2, 4, 5\}$、$sub_1 = \{0, 3, 6\}$、$sub_2 = \{1, 2, 5\}$、$sub_3 = \{0, 3\}$、$sub_4 = \{1, 6\}$、$sub_5 = \{3, 4, 6\}$的情况，如图所示：

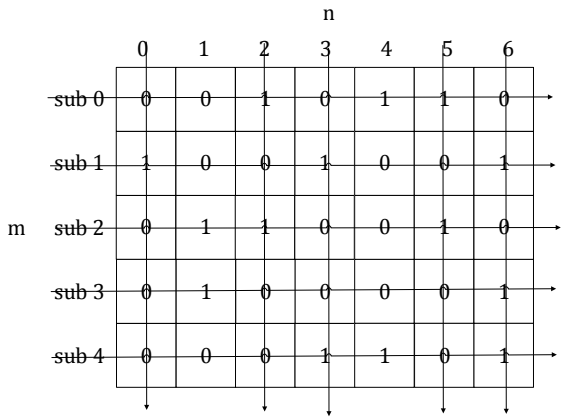
		n						
		0	1	2	3	4	5	6
m	sub 0	0	0	1	0	1	1	0
	sub 1	1	0	0	1	0	0	1
	sub 2	0	1	1	0	0	1	0
	sub 3	0	1	0	0	0	0	1
	sub 4	0	0	0	1	1	0	1

在这个矩阵 d 上进行回溯法（我个人认为回溯法和深度优先递归搜索的本质是一样的）即可得到精确覆盖，过程如下：

(1) 从 0 开始遍历集合 s 中每个成员，对于成员 0，遍历所有子集，找到第一个满足$d[1, 0] = 1$的子集sub_1，选择该子集作为精确覆盖中的一个子集，$sub_1 = \{0, 3, 6\}$包含的其他成员，其他子集不能再包含该子集中出现的成员，因此删掉其他包含$\{0, 3, 6\}$的子集sub_3和sub_4，将sub_1也删掉；

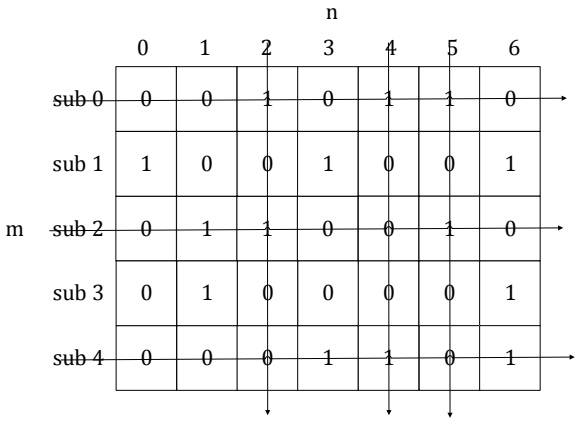


(2) 从 1 开始遍历集合 s 中剩下的成员，对于成员 1，遍历剩余子集，找到第一个满足$d[2,1]=1$的子集sub_2，选择该子集作为精确覆盖中的一个子集，$sub_2=\{1,2,5\}$包含的其他成员，其他子集不能再包含该子集中出现的成员，因此删掉其他包含$\{1,2,5\}$的子集sub_0，将sub_2也删掉，这时矩阵 d 已经被删空，并且所有成员都被包含了，说明已经找到了精确覆盖，即为$\{sub_1,sub_2\}$，算法结束；



考虑一种失败的情况：

(1) 第 1 次选择的子集为$sub_0=\{2,4,5\}$，删除包含这些成员的子集sub_2、sub_4；

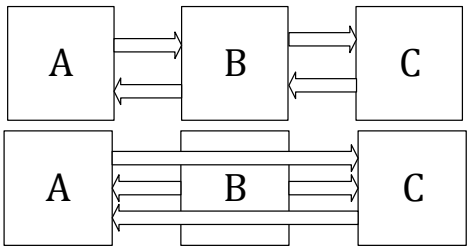


(2) 第 2 次选择$sub_1=\{0,3,6\}$，删除sub_3，这时矩阵 d 已经是空矩阵了，但还剩$\{1\}$没有包含，因此选择$\{sub_0,sub_1\}$失败了。这时需要恢复上一次删除的sub_1和sub_3，选择sub_1的理由是其包含$\{0\}$，此时继续寻找下一个包含$\{0\}$的子集，无法找到，则继续恢复上一次删除的sub_0、sub_2、sub_4（这时 d 已经恢复为原始矩阵了），选择sub_0

的理由是其包含{2}，寻找下一个包含{2}的子集sub_2，在sub_2的基础上可以进行下一波循环，最终找到正确的精确覆盖，算法结束；

回溯法的递归结束条件是矩阵 d 为空（当矩阵 d 为空时递归结束），每次递归时选择矩阵中的一列x_j（其中$0 \le j < n$），遍历矩阵 d 中的所有行，找到一行sub_i（其中$0 \le i < m$）满足$d[i,j] = 1$，选择该行。由于精确覆盖的要求，其他包含该行中成员的行，都不能再选择，因此将所有包含该行成员的其他行删掉，然后将sub_i行删掉。重复这个操作试图将矩阵 d 删空，并且在矩阵 d 变为空矩阵的时候恰好所有成员也都被已经选择的子集覆盖到。在选取包含成员x_j的行时，可能有多个选择，若选择其中一个子集而无法将 d 删空，则在递归函数中返回这一层，重新尝试下一个子集。

十字链表是一种方便删除矩阵 d 中的行列、以及恢复行列的数据结构。每个节点有上下左右 4 个指针指向周围的节点。如图所示，删除节点 B 时只需要重置$A \to B$为$A \to C$，重置$C \to B$为$C \to A$，而不需要删除$B \to A$和$B \to C$的链接：



恢复节点 B 时只需要将 B 插入其指针指向的两个节点 A 和 C 之间即可。

舞蹈链算法在最坏情况下的时间复杂度与递归的时间复杂度一样，为$O(n \times m)$。