1. 构造识别下列语言的 DFA，画出状态图，$\Sigma=\{0,1\}$。（每小题 10 分，共 40 分）

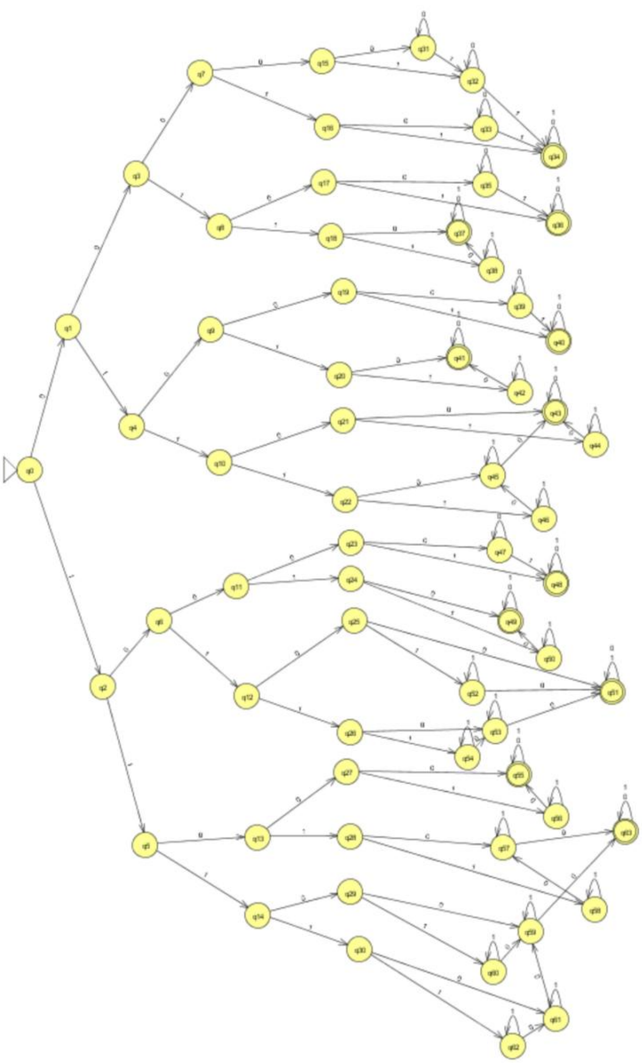
   a) $\{\ \boldsymbol{\omega} \mid \boldsymbol{\omega} \in \Sigma^*$，$\boldsymbol{\omega}$含有至少 3 个 0 和至少 2 个 1$\}$

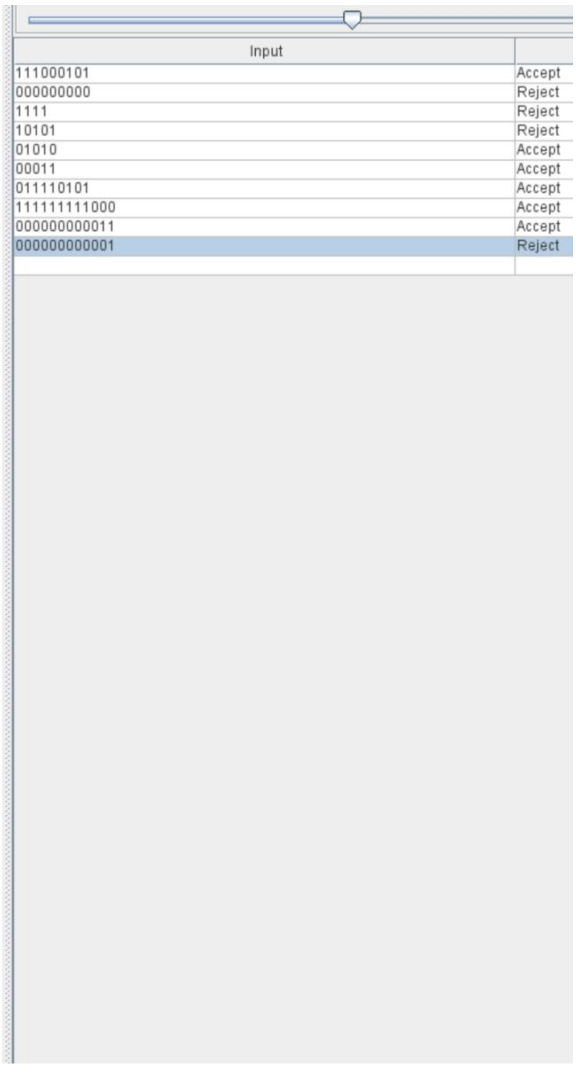
   b) $\{\ \boldsymbol{\omega} \mid \boldsymbol{\omega} \in \Sigma^*$，$\boldsymbol{\omega}$含有奇数个 0 并且以 1 结尾$\}$

   c) $\{\ \boldsymbol{\omega} \mid \boldsymbol{\omega} \in \Sigma^*$，$\boldsymbol{\omega}$中不含子串 01$\}$

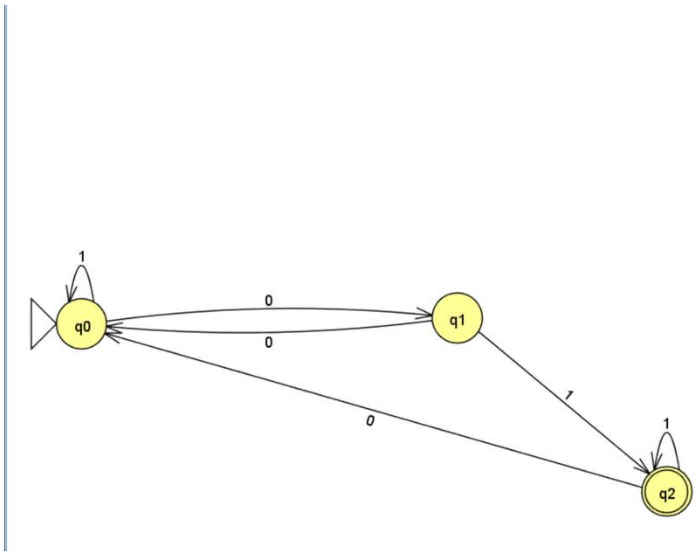
   d) $\{\ \boldsymbol{\omega} \mid \boldsymbol{\omega} \in \Sigma^*$，$\boldsymbol{\omega}$是不含 2 个 0 的任意串$\}$

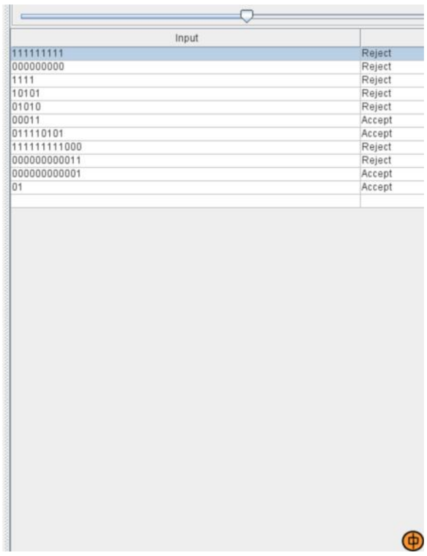
（a）考虑使用二叉树存储输入字符串的状态，五位相当于最基础的情况，因此需要建立起五位字符串之间的转移关系，比如 00000 应该如何转移，此时若下一位为 1 那么久相当于他的下面紧邻的状态，如果为 0 那么维持原态。接受态其实可以单拎出来，但做的过程中没有考虑到，因此接受态混杂进了所有的叶子状态（除了最下面的一部分，单独创建了一个接受态）。到达接受态即可停止，后续不再影响状态。





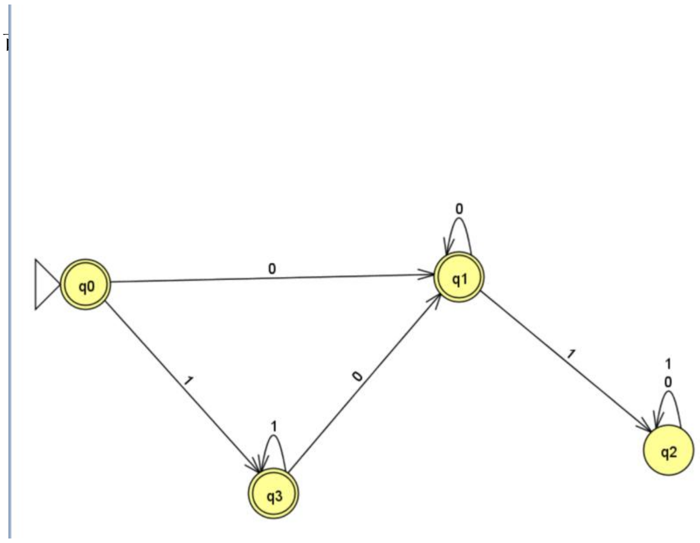
（b）直接做就行

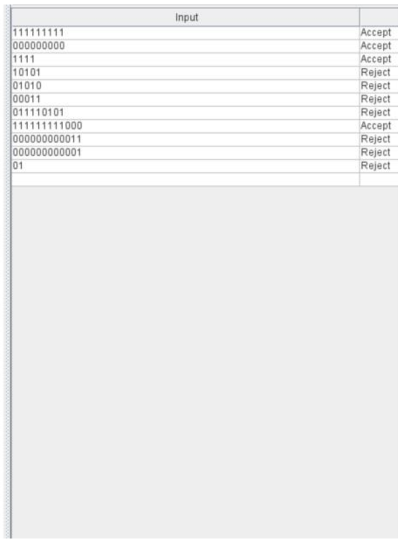




Input	
111111111	Reject
000000000	Reject
1111	Reject
10101	Reject
01010	Reject
00011	Accept
011110101	Accept
111111111000	Reject
000000000011	Reject
000000000001	Accept
01	Accept

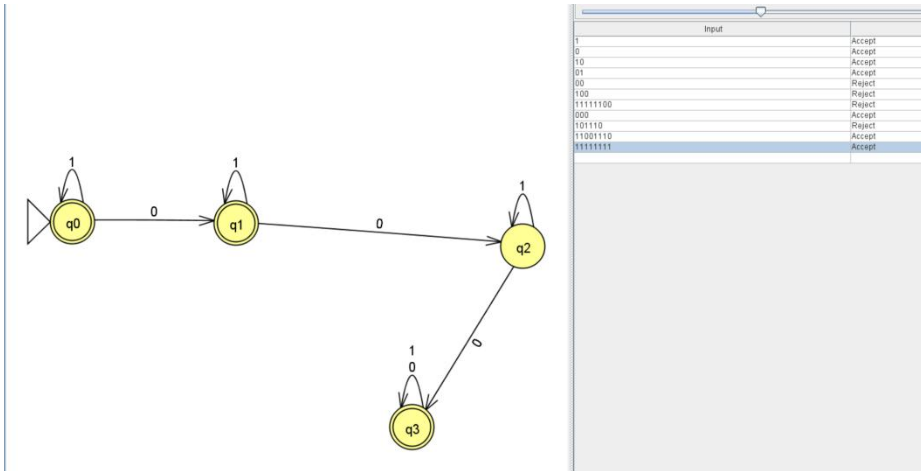
（c）逆向思维，先考虑找出 01 子串的自动机是什么样的然后将拒绝态与接受态交换即





Input	
111111111	Accept
000000000	Accept
1111	Accept
10101	Reject
01010	Reject
00011	Reject
011110101	Reject
111111111000	Accept
000000000011	Reject
000000000001	Reject
01	Reject

（d）题目的意思是 0 的个数不为 2，那么 0 的个数可以是大于等于 3 或者是 0 或 1，直接做就行。



2. 构造识别下列语言的 NFA，画出状态图，Σ={0,1}。(每小题 10 分，共 40 分)

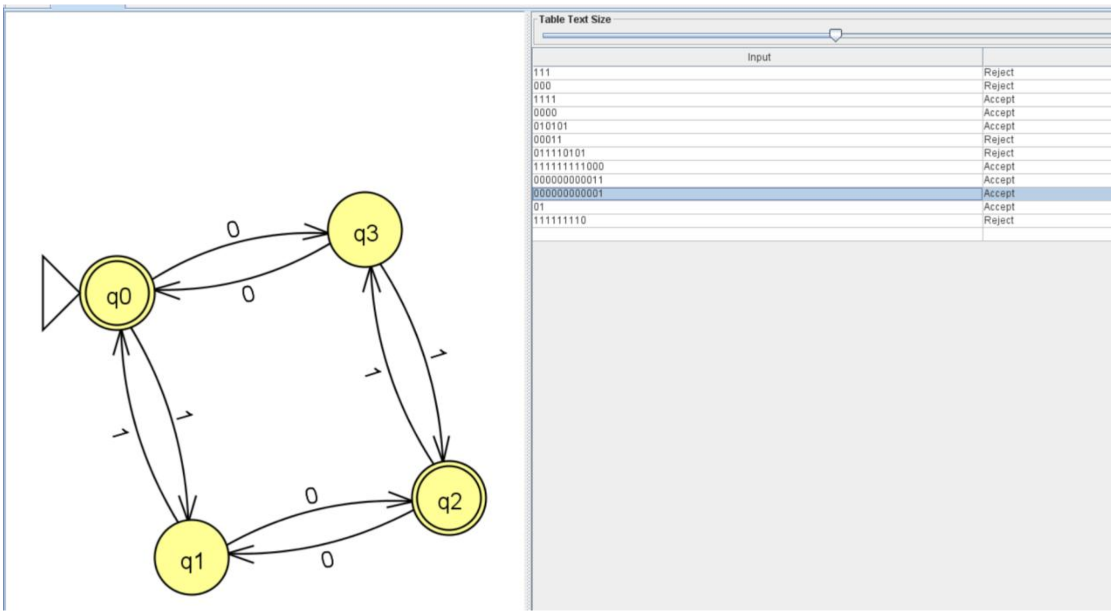
   a) { $\boldsymbol{\omega}|\boldsymbol{\omega}\in\Sigma^*$，0 和 1 的个数要么都是偶数，要么都是奇数}

   b) { $\boldsymbol{\omega}|\boldsymbol{\omega}\in\Sigma^+$，$\boldsymbol{\omega}$中不含形如 10110 的子串}

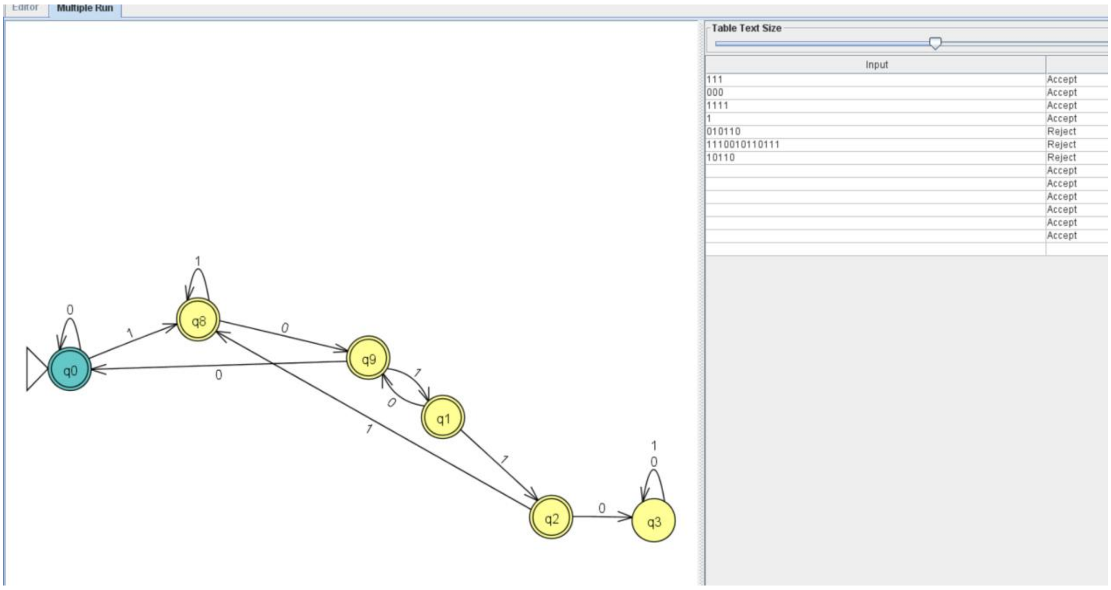
   c) { $\boldsymbol{\omega}|\boldsymbol{\omega}\in\Sigma^*$，如果$\boldsymbol{\omega}$以 1 结尾，则它的长度为偶数；如果$\boldsymbol{\omega}$以 0 结尾，则它的长度为奇数}

   d) { $\boldsymbol{\omega}|\boldsymbol{\omega}\in L$，L= $\{1^{2n} \cup 1^{3n}|n \geqslant 1\}$ }

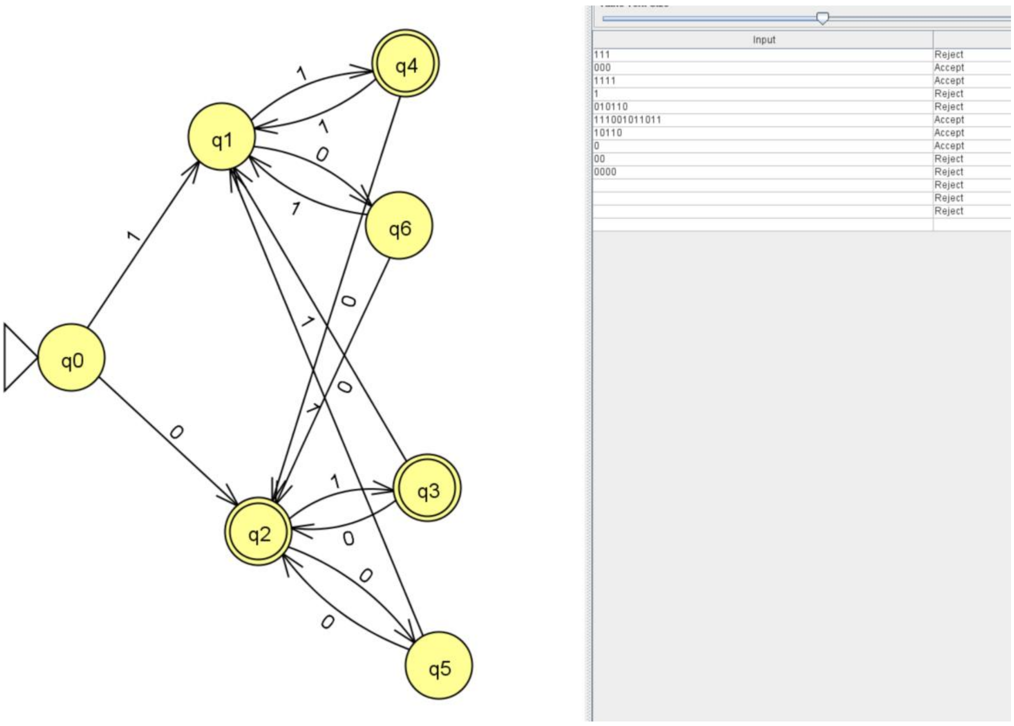
（a）q0 开始，不管是输入了 0 还是 1 都不是接受态，但是如果再输入了 0 或 1 都是接受态，q0 与 q2 代表了偶数个 01 和奇数个 01，q3 代表了奇数个 0 偶数个 1，q1 代表了奇数个 1 偶数个 0，这里直接给出我最先想的答案。



（b）与之前的思路一样，都是逆向思维，先构造出接受子串的 DFA 然后“取反”就行。

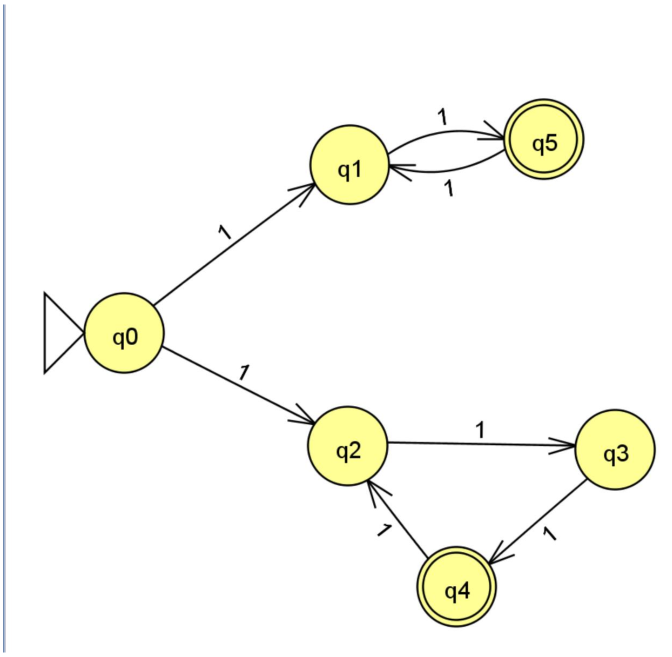


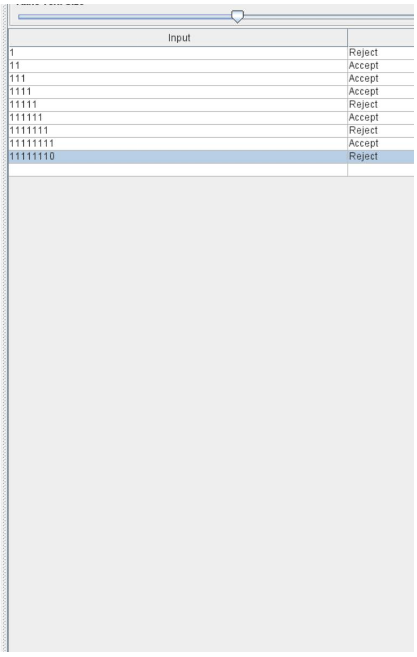
（c）从最基础的情况下手，比如 0,1,00,11,10,11，这些情况搞清楚了之后再去考虑他们接下来都会怎么转移，比如 10，如果是 1 的话那就是 101，意味着此时以 1 结尾但是长度为奇数，对应的状态是 q1 这个状态，100 的话意味着此时以 0 结尾并且长度为奇数，此时对应的是 q2 这个状态。只要把这些基础情况搞好了其他再怎么变也就是只在增加长度罢了。



（d）按照上课的例题，选择两路进行构造，分别代表 2 的倍数和 3 的倍数。然后直接

构造。

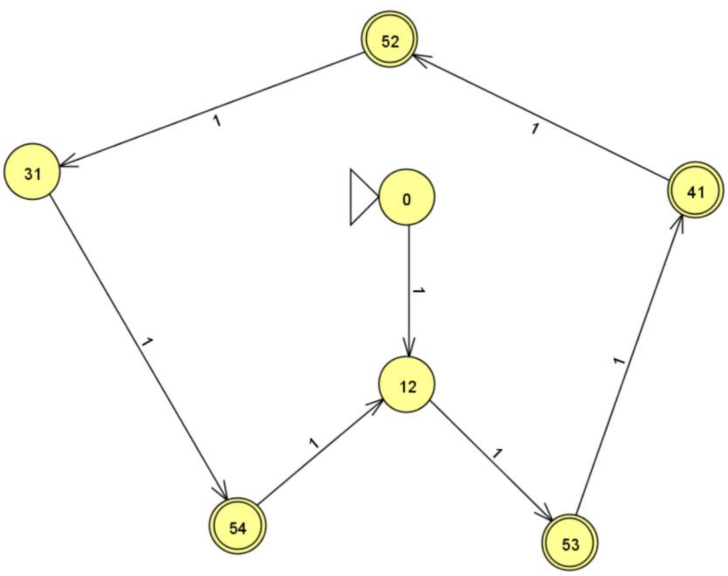


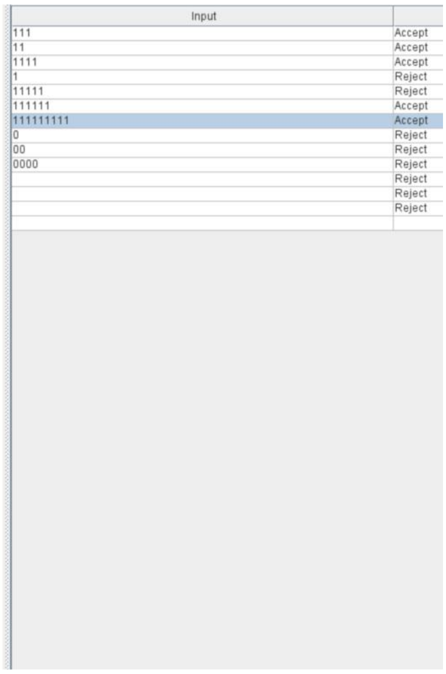


Input	
1	Reject
11	Accept
111	Accept
1111	Accept
11111	Reject
111111	Accept
1111111	Reject
11111111	Accept
11111110	Reject

3. 将习题 2.d 的 NFA 转化为等价的 DFA。（本题 20 分）

按照老师上课讲的方法 2（基于关键字集合设计 NFA 的特殊子集构造法）进行构造，节点的名字对应 NFA 的各个状态。本题特别强调了 w 属于 L 因此不用加入处理陷进状态的另一个状态。

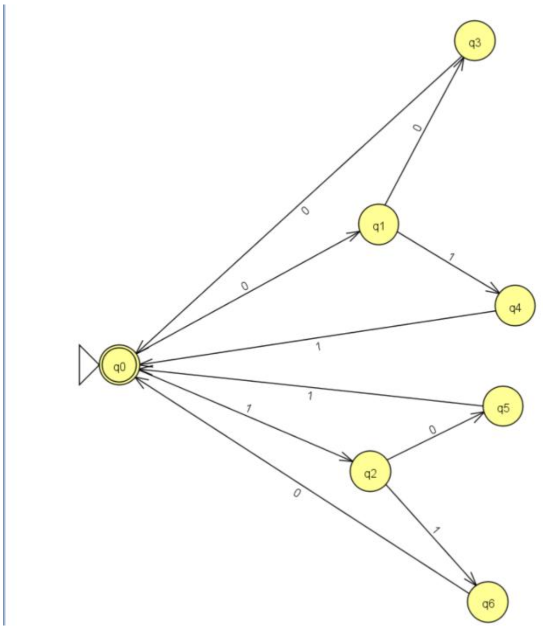




Input	
111	Accept
11	Accept
1111	Accept
1	Reject
11111	Reject
111111	Accept
111111111	Accept
0	Reject
00	Reject
0000	Reject
	Reject
	Reject
	Reject

4. 语言 L={$\boldsymbol{\omega}$ | $\boldsymbol{\omega}$ =$a_0b_0c_0a_1b_1c_1\cdots a_nb_nc_n$，$\begin{array}{r} a_n\ldots a_1a_0 \\ +\quad b_n\ldots b_1b_0 \\ \hline c_n\ldots c_1c_0 \end{array}$，$a_i,b_i,c_i\in\{0,1\}$，$n\geqslant 0$，$0\leqslant i\leqslant n$，这里的加号“+”代表二进制加},试判断 L 是不是正则语言。如果是的话，请构造识别该语言的 DFA。（可选题，100 分）

如果我们考虑的二进制加法是不进位的，那么我们就可以构造出如下的 DFA：如果能构造出 DFA 的话，那么 L 一定就是正则语言。这里我们假设的二进制加法是异或。



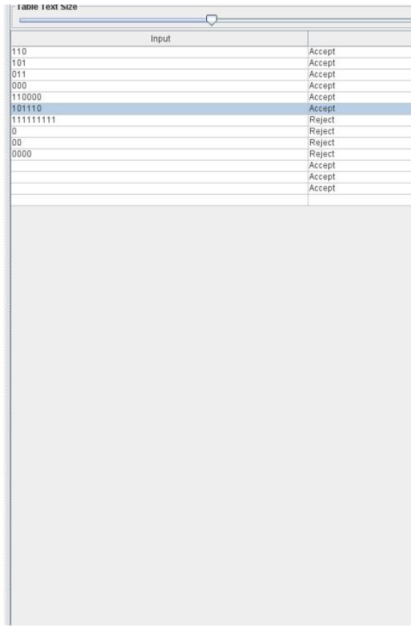


Table Text Size

Input	
110	Accept
101	Accept
011	Accept
000	Accept
110000	Accept
101110	Accept
111111111	Reject
0	Reject
00	Reject
0000	Reject
	Accept
	Accept
	Accept

下面思考如果考虑进位呢？这里我们假设如果做完加法之后发生溢出,那么这种加法也是非法的，比如 1111+1111=11110，最高位发生溢出，是非法，而如果是 01111+01111=11110 的话没有发生溢出，那么这样的就是合法的。因此需要考虑 1+1=10 这样的溢出情况，以竖式的角度来讲，a=1，b=1，c 一定是 0，那么我们只需要对上图的一部分进行修改即可。

下面考虑 2 位二进制数的加法，如下图所示：

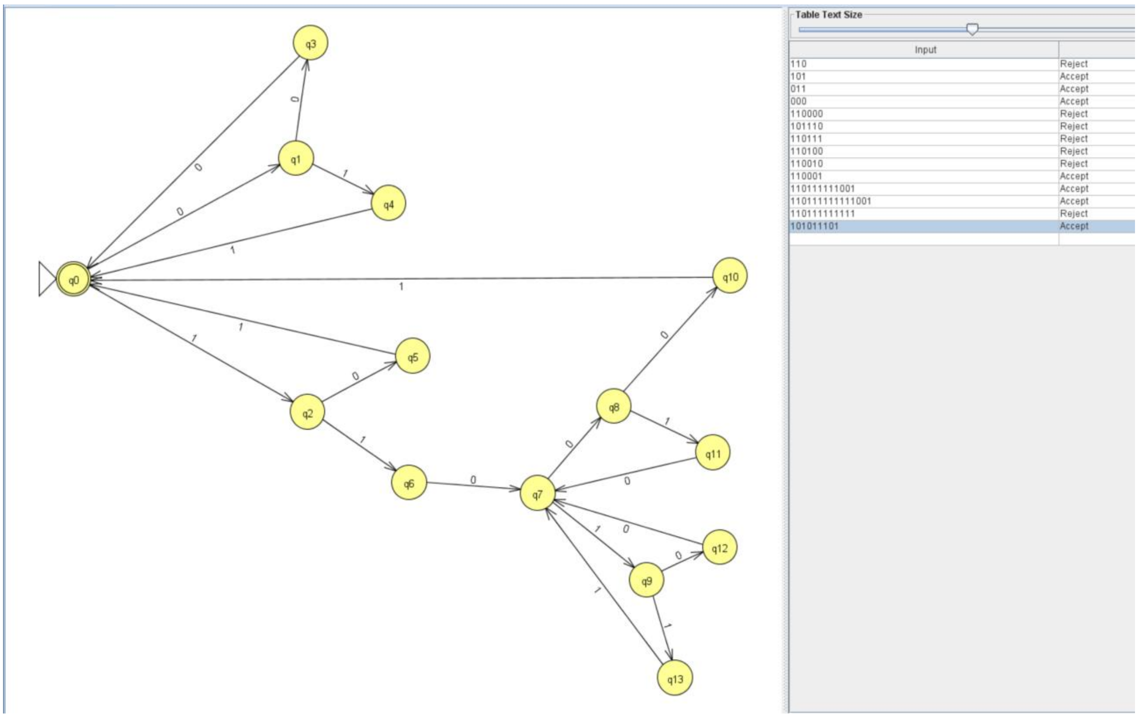
①
11
11
110
110111 非

②
11
01
100
110100 非

③
01
11
100
110010 非

④
01
01
10
110001 合法

123 三种情况都可以认为发生了进位，因此可以将他们归为一类，只有第 4 种情况属于合法情况，因此构造出如下的 DFA：其中 q11、q12、q13 都可以认为是一类也就是发生了进位的情况，在我们的假设下这种情况是非法的，并且他们都转移到 q7，q7 也相当于一个发生了进位的非法情况。（如果需要修改的话将 q7 改为接受态就可以）



5. （寻宝谜题）Alice 在一个藏宝山洞门口。山洞有一个门，门上有 n 个硬币，它们排列成一个圈，Alice 无法知道它们的正反情况，每次只能翻转其中的 k（k≤ n）个，且每次翻转之后，这些硬币会随机顺时针旋转（硬币之间的相对位置不变），随机旋转的角度未知。已知只有将这些硬币全部翻成正面或全部翻成反面才能打开这扇门。你能不能帮 Alice 想一个办法，使得硬币无论每次怎样旋转，一定能在某一限定操作次数内打开这扇门？（可选题，100 分）

首先从自动机的角度思考这个问题，我们可以把硬币的状态看作由 0 和 1 构成的字符串，其中 0 表示反面，1 表示正面。因为硬币排成一个环形，旋转操作可以等价为这个 01 字符串的首尾相接平移。我们的目标是找到一种翻动硬币的策略，使得无论对手如何旋转，最终一定能达到所有硬币同面（全 0 或全 1）的状态。

旋转这一动作虽然使得我们无法确定每次操作后的确切位置，但它不会改变硬币之间的相对位置，因此从自动机的角度看，旋转后的状态数是有限的。例如，当有三个硬币时，如果一次翻转后的状态为 101，那么旋转后可能的状态只会是 101、011 或 110。这意味着整个过程可以用一 NFA 来刻画：每个状态对应一个 01 串，而每个翻转操作对应若干可能的后继状态（所有旋转等价类）。

存在必胜策略的条件：当且仅当 n 和 k 互质时，Alice 存在一个与初始状态和旋转无关的必胜策略。若 (gcd(n, k) > 1)，环被分成若干互不影响的等距类，可能始终保持部分硬币面相反，因此无法保证收敛到全同状态。举出反例：比如 n=6，k=4，那么如果是 010101 的字符串的话，我们设 024 的位置的硬币为 A 组，135 位置的硬币为 B 组，此时 A、B 组中的 1 的个数分别为（奇，偶），最终全 0 或者全 1 状态二元组应该变为（奇，奇）或（偶，偶），但是由于每次变化都只能选择 4 个硬币，并且对手可以旋转硬币，因此会存在着一种情况也就是 A、B 组内 1 的个数永远都保持（奇，偶），最终无法成功。因此 n 与 k 互质是解答这道题的必要条件。

我们的必胜策略如下：设一次动作（Alice 的操作）为：从某个起点 i（i 在 0-n-1 之间）开始，翻转连续的 k 枚硬币（按环形取模），之后对手执行一次任意旋转。策略为：按顺序执行 (a0, a1, an-1)即依次从每个起点出发各翻一次连续的 k 枚硬币，然后再重复这一轮。对于 (n=3, k=2) 的情况，这一轮动作就是 abc（或其循环位移如 bca、cab 等），重复执行即可。

证明：定义一个“边界数” B(x)，表示当前状态串中相邻两枚硬币面不相同的边数（包括首尾）。旋转不会改变 B(x)。若翻转的那一段跨过至少一条边界，则 B 至少减少 1。比如：0101 中 B(x)=4，如果 k=3 的话，x1=1011 那么执行一次动作之后 B(x1)=2。

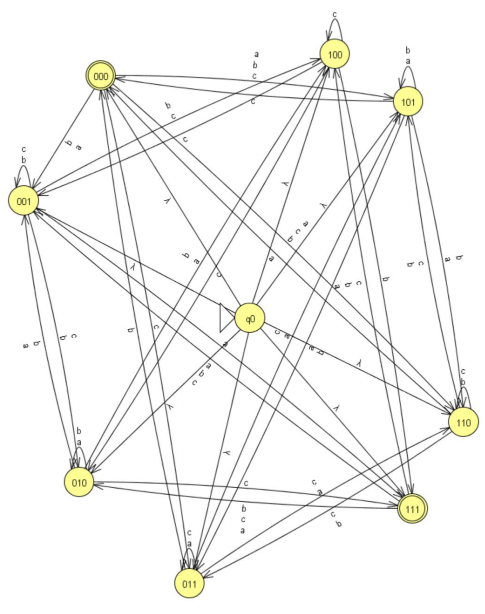
当 gcd(n,k)=1 时，所有起点构成一个完整的剩余系。剩余系是数论中的一个基本概念：对于模 n，一组整数若能保证它们在模 n 的意义下两两不同（即每个可能的余数 0,1,⋯,n−1 都恰好出现一次），那么这组数称为模 n 的一个完全剩余系。

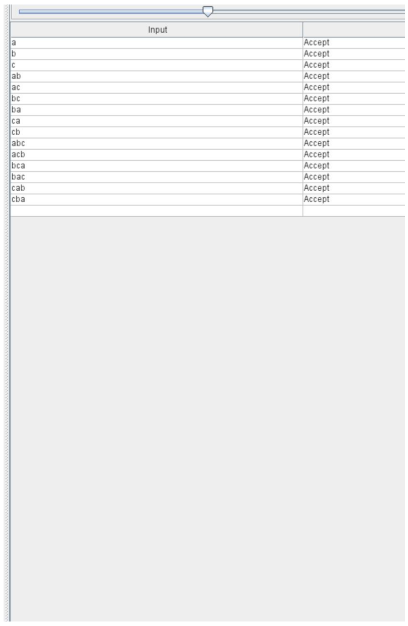
在本题中，Alice 每次的操作是从起点 i 开始翻转连续的 k 枚硬币（按环取模）。当 gcd(n,k)=1 时，起点 0,1,2,⋯,n−1 经过不断地加 k 取模运算，会依次覆盖所有可能的位置而不重复，这就形成了一个模 n 的完全剩余系。通过不断平移起点(即依次执行 a0,a1,⋯,an−1，Alice 的操作能扫过环上所有可能的相对位置。

因此，无论对手怎样旋转，Alice 在一整轮动作中（执行完从 a0 到 an−1）必有一次翻转能覆盖到“真实的边界”，从而使得边界数 B 减少至少 1。由于初始时 B≤n，经过有限（最多 n 轮）操作后，必然有 B=0 也就是所有硬币面相同（全 0 或全 1），门被打开。

旋转的不可控性被自动机中的“非确定性”吸收，Alice 的策略等价于对整个状态空间的穷举覆盖。当 gcd(n,k)=1 时，这样的自动机必然收敛到接受状态（全 0 或全 1）。因此，上述策略即为通用的必胜方案。

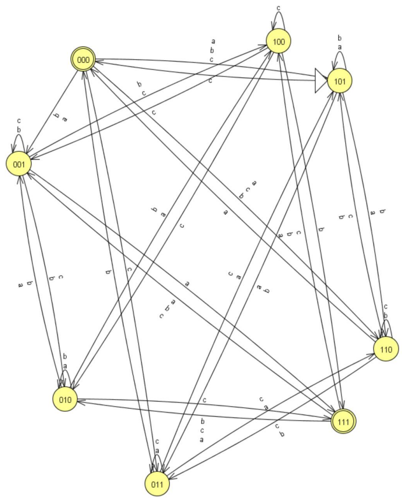
以下给出自动机的验证结果，假设 k=2，n=3。



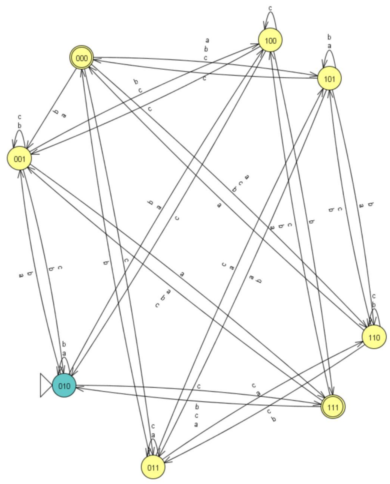


Input	
a	Accept
b	Accept
c	Accept
ab	Accept
ac	Accept
bc	Accept
ba	Accept
ca	Accept
cb	Accept
abc	Accept
acb	Accept
bca	Accept
bac	Accept
cab	Accept
cba	Accept

以下给出一些初始态给定的情况作为验证：可见也不是所有的动作序列都可以完成目标，但一定存在某种动作序列可以完成目标。



Input	
a	Reject
b	Reject
c	Accept
ab	Accept
ac	Accept
bc	Accept
ba	Accept
ca	Reject
cb	Reject
abc	Accept
acb	Accept
bca	Accept
bac	Accept
cab	Reject
cba	Accept



Input	
a	Reject
b	Reject
c	Accept
ab	Accept
ac	Accept
bc	Accept
ba	Accept
ca	Reject
cb	Reject
abc	Accept
acb	Accept
bca	Accept
bac	Accept
cab	Accept
cba	Accept