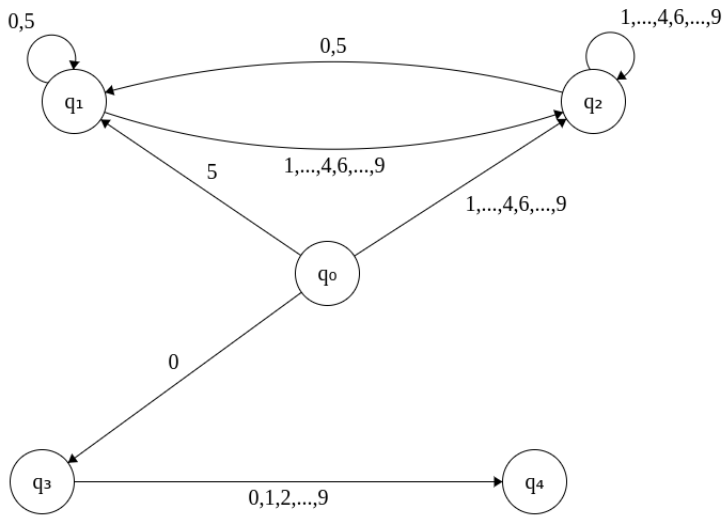
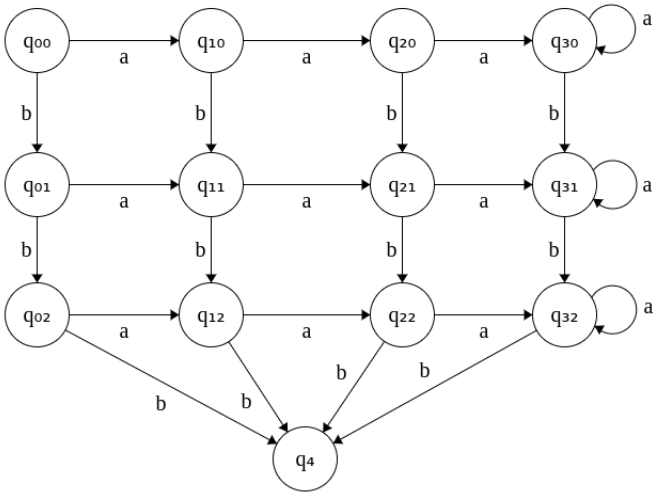
1



Здесь q_4 – заземление, а q_3 и q_1 являются терминальными.

2



Здесь q_4 – заземление, а q_{30}, q_{31}, q_{32} являются терминальными.

3

Вот что я узнал по поводу Python (мой любимый язык после плюсов):

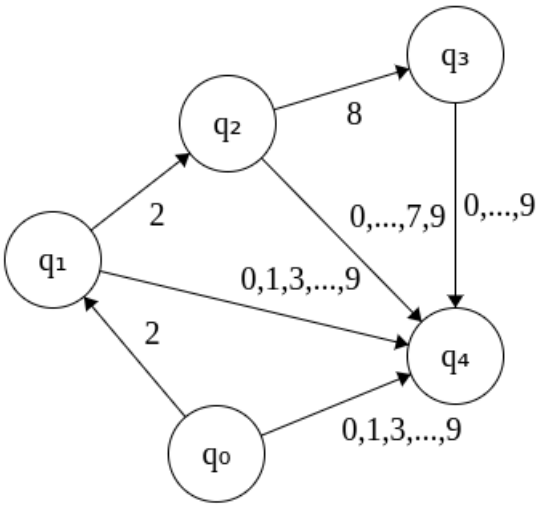
- Можно переносить строки, не разъединяя токены, с помощью \
- Можно с помощью комментария в начале файла указывать кодировку

Источник

4

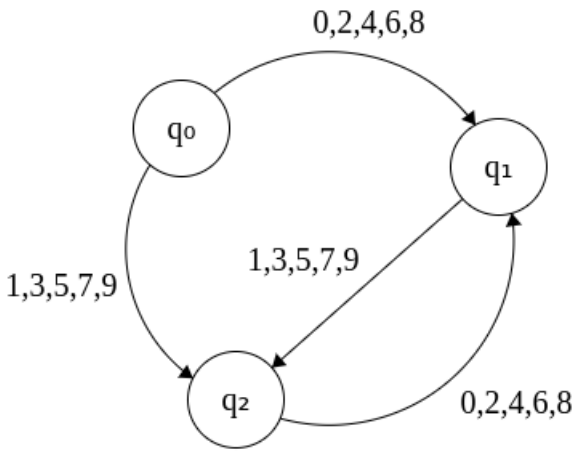
Заметим по сути, что каждый автомат – это граф, тогда будем с помощью алфавита из цифр, запятой, точки с запятой, восклицательного знака и собаки. Будем задавать граф следующим образом: количество вершин, восклицательный знак, номер вершины и её тип в формате num,1010; , где четыре цифры отвечают за тип следующим образом: если правда, то что вершина начальная – то первая цифра 1, иначе 0, если вершина обычная, то соответственно, если вершина терминальная, то соответственно и если вершина является заземлением, то аналогично. Далее, после всех вершин идёт ещё @. Потом каждое ребро идет в формате: "[номер первой вершины],[номер второй вершины],[перечисление через запятую всех чисел по которым мы переходим];". Примеры: автомат из первого задания будет выглядить так: $5!0,1000;1,0100;2,0010;3,0010;4,0001;@1,1,0,5;0,1,5;0,2,1,2,3,4,6,7,8,9;2,2,1,2,3,4,6,7,8,9;1,2,0,5;2,1,1,2,3,4,6,7,8,9;0,3,0;3,4,0,1,2,3,4,5,6,7,8,9;$

Теперь возьмём что-нибудь попроще, автомат, проверяющий, не дали ли ему на вход число 228:



Записывается так: $5!0,1000;1,0100;2,0100;3,0010;4,0001;@0,1,2;0,4,0,1,2,3,4,5,6,7,8,9;1,2,2;1,4,0,1,3,4,5,6,7,8,9;2,3,8;2,4,0,1,2,3,4,5,6,7,9;3,4,0,1,2,3,4,5,6,7,8,9;$

Возьмем что-нибудь еще проще: автомат проверяющий, является ли число чётным:



Записывается так: $3!0,1000;1,0010;2,0010;@0,1,0,2,4,6,8;0,2,1,3,5,7,9;1,2,1,3,5,7,9;2,1,0,2,4,6,8;$

5

Вот пример того, что получилось (поменял в файлах репозитория конфиг на свой):