1 Задание 6

1.1 Задача 1

Кажется, что да, попробуем это доказать.

Единица на нечетном месте, будем считать слева направо, при делении на 3 даёт остаток 1, на четном даёт остаток 2:

$1 \bmod 3 \equiv 1$, $2 \bmod 3 \equiv = 2$, $4 \bmod 3 \equiv = 1$.

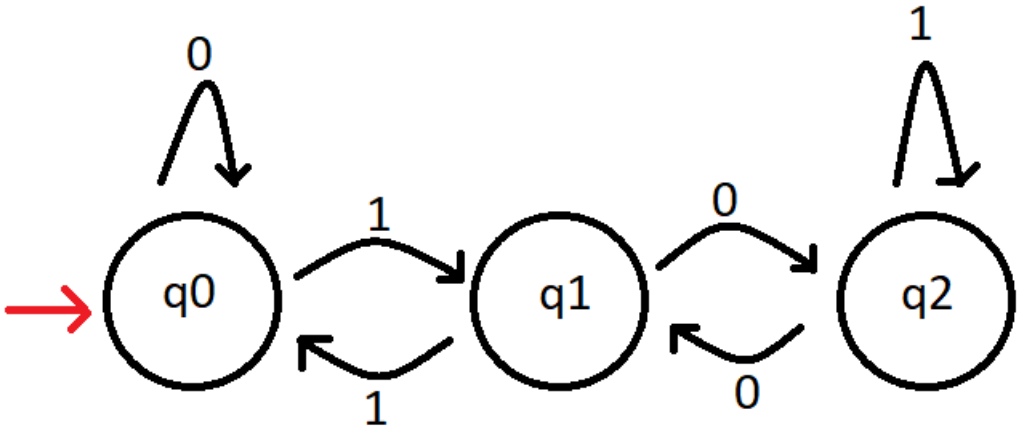
При добавлении символа каждая четная позиция становится нечетной, и каждая нечетная становится четной. Новый символ (слева) меняет позиции с четных на нечетные и нечетные на четные, а сам считываемый символ всегда на несчитываемой символ. Тогда будем рассматривать следующие комбинации: Кол-во Ч (Остаток от деления кол-ва единиц на четных местах), Кол-во Н (Остаток от деления кол-ва единиц на нечетных местах), s – слово, которое будем дописывать.

N	Кол-во Ч	Кол-во Н	$\omega \bmod 3$	s	ω·s $\bmod 3$
1	1	1	$1 \cdot 2 + 1 \cdot 1 \equiv 0 \bmod 3$	0	$1 \cdot 2 + 1 \cdot 1 + 0 \equiv 0 \bmod 3$
2	1	2	$1 \cdot 2 + 2 \cdot 1 \equiv 1 \bmod 3$	0	$2 \cdot 2 + 1 \cdot 1 + 0 \equiv 2 \bmod 3$
3	1	0	$1 \cdot 2 + 0 \cdot 1 \equiv 2 \bmod 3$	0	$0 \cdot 2 + 1 \cdot 1 + 0 \equiv 1 \bmod 3$
4	2	1	$2 \cdot 2 + 1 \cdot 1 \equiv 2 \bmod 3$	0	$1 \cdot 2 + 2 \cdot 1 + 0 \equiv 1 \bmod 3$
5	2	2	$2 \cdot 2 + 2 \cdot 1 \equiv 0 \bmod 3$	0	$2 \cdot 2 + 2 \cdot 1 + 0 \equiv 0 \bmod 3$
6	2	0	$2 \cdot 2 + 0 \cdot 1 \equiv 1 \bmod 3$	0	$0 \cdot 2 + 2 \cdot 1 + 0 \equiv 2 \bmod 3$
7	0	1	$0 \cdot 2 + 1 \cdot 1 \equiv 1 \bmod 3$	0	$1 \cdot 2 + 0 \cdot 1 + 0 \equiv 2 \bmod 3$
8	0	2	$0 \cdot 2 + 2 \cdot 1 \equiv 2 \bmod 3$	0	$2 \cdot 2 + 0 \cdot 1 + 0 \equiv 1 \bmod 3$
9	0	0	$0 \cdot 2 + 0 \cdot 1 \equiv 0 \bmod 3$	0	$0 \cdot 2 + 0 \cdot 1 + 0 \equiv 0 \bmod 3$
10	1	1	$1 \cdot 2 + 1 \cdot 1 \equiv 0 \bmod 3$	1	$1 \cdot 2 + 1 \cdot 1 + 1 \equiv 1 \bmod 3$
11	1	2	$1 \cdot 2 + 2 \cdot 1 \equiv 1 \bmod 3$	1	$2 \cdot 2 + 1 \cdot 1 + 1 \equiv 0 \bmod 3$
12	1	0	$1 \cdot 2 + 0 \cdot 1 \equiv 2 \bmod 3$	1	$0 \cdot 2 + 1 \cdot 1 + 1 \equiv 2 \bmod 3$
13	2	1	$2 \cdot 2 + 1 \cdot 1 \equiv 2 \bmod 3$	1	$1 \cdot 2 + 2 \cdot 1 + 1 \equiv 2 \bmod 3$
14	2	2	$2 \cdot 2 + 2 \cdot 1 \equiv 0 \bmod 3$	1	$2 \cdot 2 + 2 \cdot 1 + 1 \equiv 1 \bmod 3$
15	2	0	$2 \cdot 2 + 0 \cdot 1 \equiv 1 \bmod 3$	1	$0 \cdot 2 + 2 \cdot 1 + 1 \equiv 0 \bmod 3$
16	0	1	$0 \cdot 2 + 0 \cdot 1 \equiv 1 \bmod 3$	1	$1 \cdot 2 + 0 \cdot 1 + 1 \equiv 0 \bmod 3$
17	0	2	$0 \cdot 2 + 2 \cdot 1 \equiv 2 \bmod 3$	1	$2 \cdot 2 + 0 \cdot 1 + 1 \equiv 2 \bmod 3$
18	0	0	$0 \cdot 2 + 0 \cdot 1 \equiv 0 \bmod 3$	1	$0 \cdot 2 + 0 \cdot 1 + 1 \equiv 1 \bmod 3$

Остатки при делении на 3 задают классы эквивалентности по отношению $\sim$ действительно:

1. $\omega \bmod 3 \equiv 1$, то $\omega \cdot 0 \bmod 3 \equiv 2 \in \mathrm{L}$, $\omega \cdot 1 \bmod 3 \equiv 0 \notin \mathrm{L}$
2. $\omega \bmod 3 \equiv 2$, то $\omega \cdot 0 \bmod 3 \equiv 1 \notin \mathrm{L}$, $\omega \cdot 1 \bmod 3 \equiv 2 \in \mathrm{L}$
3. $\omega \bmod 3 \equiv 0$, то $\omega \cdot 0 \bmod 3 \equiv 0 \notin \mathrm{L}$, $\omega \cdot 1 \bmod 3 \equiv 1 \notin \mathrm{L}$

Построим ДКА из этих классов по алгоритму и получим:



Доказано

1.2 Задача 2

а) cooming soon... **б)** cooming soon... **в)** cooming soon...

1.3 Задача 3

Угадали решение: $n = 3 \cdot k + 1$ (на самом деле нашли с помощью алгоритма Евклида)

Найдем классы L-эквивалентности. Классы эквивалентности найдены аналогично задачи 1:

1. L_0 – язык всех слов, дающих остаток 0 при целочисленном делении на 3
2. L_1 – язык всех слов, дающих остаток 1 при целочисленном делении нв 3
3. L_2 – язык всех слов, дающих остаток 2 при целочисленном делении на 3

Получили, что язык L – регулярен.
Ответ: Язык L – регулярен.

1.4 Задача 4

а) Пусть L = {xy : |x| > |y|, x содержит букву a}

Выберем два множества: L_1 – язык всех слов, не содержащих букву a, L_2 – язык всех слов, содерщащих букву a.

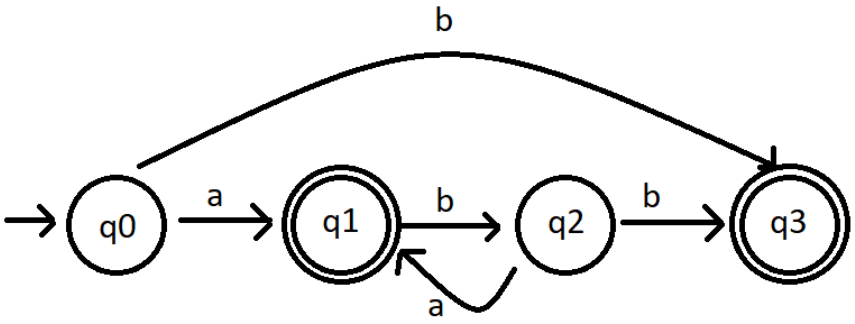
Теперь будет проверять будут ли они классам L-эквивалентности. Возьмём два любых слова из L_1 и L_2, тогда приписывание произвольного слова $t \in \sum^*$ меняет принадлежность к L получшихся слов одновременно.

Пусть теперь $x \in L_1$, $y \in L_2$ и $\omega = b$ будет разделяющим словом. При приписывании слова t к словам x и y, получаем, что $yt \in L$, $xt \notin L$. Получили два класса L-эквивалентности. По теореме Майхилла-Нероуда L – регулярный язык.
б) cooming soon...

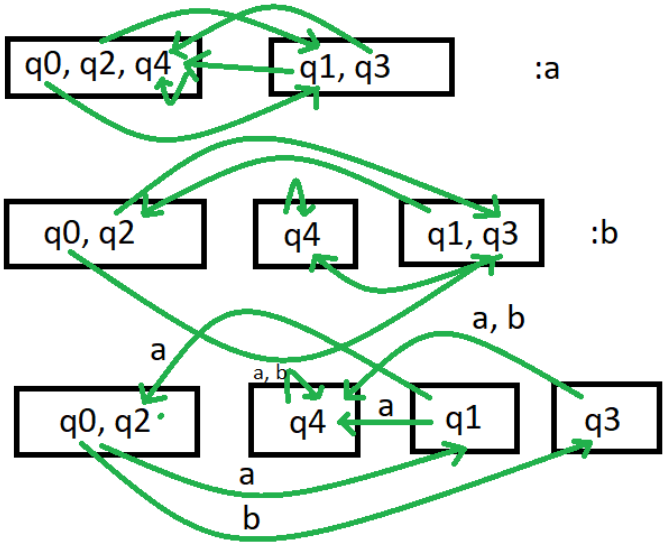
1.5 Задача 5

а) Рассмотрим два произвольных слова из языка L_q. Количество состояний конечно и определены переходы по всем буквам алфавита, т.к. $\mathcal{A}$ - полный ДКА. Можем приписать к двум любым словам из L_q ω. Подадим получившиеся слово на вход в автомат, тогда он придет сначала в состояние q, это следует из определения L_q. Потом автомат приступит к обработке слова ω, т.к. автомат ДКА то он придет либо в принимающее состояние, либо в непринимающее состояние. Т.е. все слова из левого языка будут принадлежать некоторому «левому» классу L – эквивалентности.
Доказано
б) Для каждой вершин вершины автомата $\mathcal{A}$ рассмотрим множество L_q. Теперь рассмотрим L-эквивалентность слов из разных L_q. Если у нас два слова окажутся эквивалентными, то в далее будем рассматривать их вместе. Таким образом получим разбиение множества всех «левых» языков L_q для каждой вершина на множества L_{q_k}, которые будут попарно неэквивалентны в силу построения. Т.е. для каждого класса эквивалетности существует подмножество состояние, что этому классу соответствует язык L_q, который мы можем представить в виде объединения L_{q_k} по всем состояниям из рассматриваемого множества состояний.
Доказано
в) Если у нас есть два слова из классов L_q и L_t и они лежат в одном классе эквивалентности, то при дописывании к ним любого слова ω означает то, что автомат закончит работу в одинаковом состоянии. Если так получилось, что автомат закончил в принимающеем состоянии, тогда это означает то, что $R_q = R_p$, иначе получаем, что эти два слова не принадлежат одному и тому же классу эквивалентности.
Доказано

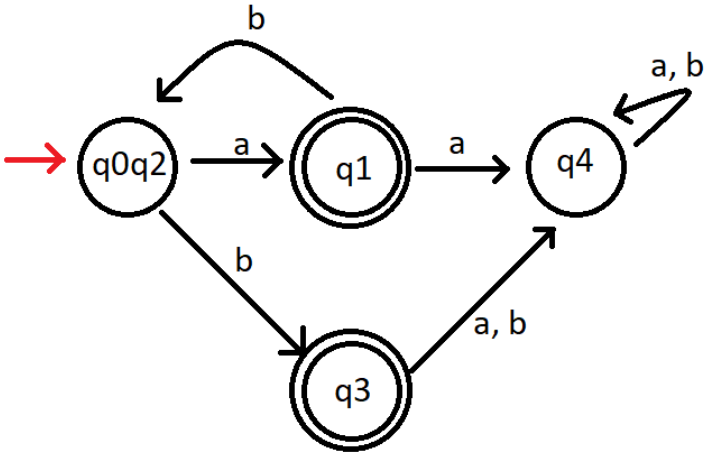
1.6 Задача 6



Построим табличку непринимающих состояний и принимающих состояний и выполним алгоритм.



Ура, всё получилось, теперь можем построить ДКА из этой картинки.



1.7 Задача 7

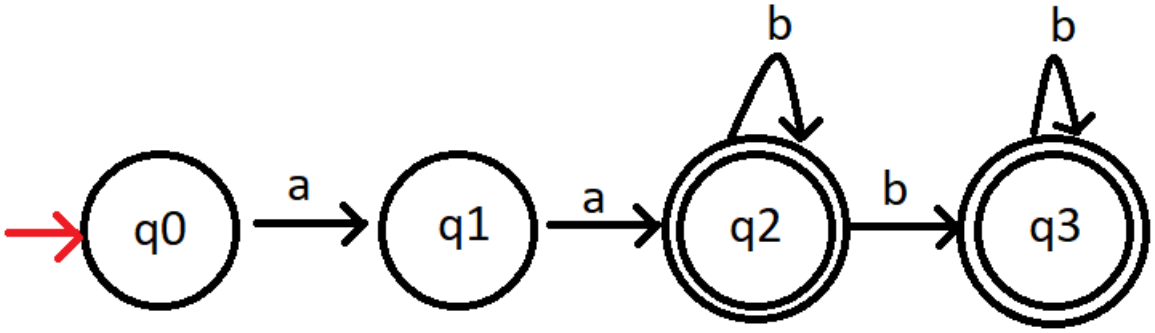
L = $\{ab^{2^i} \mid i \geqslant 0\} \cup \{b^j \mid j \geqslant 0\} \cup \{a^m b^n \mid m > 1, n \geqslant 0\}$
$L_1 = \{ab^{2^i} \mid i \geqslant 0\}$
$L_2 = \{b^j \mid j \geqslant 0\}$
$L_3 = \{a^m b^n \mid m > 1, n \geqslant 0\}$
$L = L_1 \cup L_2 \cup L_3$

Теперь, докажем, что лемма о накачке выполняется для этого языка L.

Рассмотрим лемму: для L_1 можем взять $p_1 = 1$, $x = \epsilon$, y = a($|y| = 1 \leqslant p_1$), z = b^{2^i}. Тогда при k = 0 $xy^k z = b^{2^i} \in L_2 \subseteq L$; При k = 1 $xy^k z = ab^{2^i} \in L_1 \subseteq L$. При k > 1 $xy^j = a^j b^{2^i} \in L_3 \subseteq L$. Значит все слова из L_1 удовлетворяют лемме о накачке для L.

Рассмотрим лемму: для L_2 можем взять $p_2 = 1$, $x = \epsilon$, y = b($|y| = 1 \leqslant p$), z = b^{j-1}. Тогда $xy^k z = b^{j+k-1} \in L_2 \subseteq L$. Значит все слова из L_2 удовлетворяют лемме о накачке для L.

Для L_3 можно построить ДКА $\mathcal{A}_3$. Приведу его ниже:



Значит для L_3 выполняется лемма о накачке: $\exists p_3 \forall \omega \in L_3 : |\omega| > p_3, \exists xyz = \omega$ $((y \neq \epsilon) \wedge (|xy| \geqslant p_L)) \wedge (\forall i \geqslant 0\ xy^i z \in L)) \rightarrow xy^i z \in L$.

Доказано

Теперь докажем, что $L \notin REG$. $L_1 \cap L_2 = \emptyset$, $L_1 \cap L_3 = \emptyset$, $L_2 \cap L_3 = \emptyset$. $L1 = L\ (L_2 \cup L_3)$. REG замкнуто относительно разности и объединения, тогда от противного предположим, что L $\in$ REG, т.к. $L_3 \in REG$ по доказанному выше, а $\{b\} \in REG \rightarrow L_2 = \{b\}^* \in REG, (L_2 \cup L_3) \in REG \rightarrow L_1 \in REG$.

Но $L_1 \notin REG$. Для него не выполняется лемма о накачке, т.е.

$\forall p \exists \omega \in L : |\omega| > p, \forall xyz = \omega((y = \epsilon) \bigvee (|xy| > p) \bigvee (\exists \geqslant 0 : xy^t z \notin L))$

Теперь попробуем показать это: Если взять $x = \epsilon$, y = ab^k, то $xy^0 z = b^{2^i - k} \notin L_1$, если взять y = b^k, то $xy^i z = ab^{2^i - k + tk} = ab^{2^i + (t-1)k}$. Если бы $L_1 \in REG$, то $\forall k > 0$ $\forall t \geqslant 0\ 2^i + (t-1)k = 2^j$. Но при четном k, i > 0 или нечетного k, i = 0 это не будет выполнено ни для какого четного t, а в случае нечетного k, i > 0 или четного k, i = 0 - ни для какого четного t.

Следовательно лемма не выполнется, т.е. $L_1 \notin REG \rightarrow L \notin REG$

Доказано