## Ахо-Корасик

<https://habrahabr.ru/post/198682/>

<http://e-maxx.ru/algo/aho_corasick>

## Детерминирование автомата

Заведем очередь.  
Будем класть туда множества состояний.  
Сначала там лежит множество всех входных.

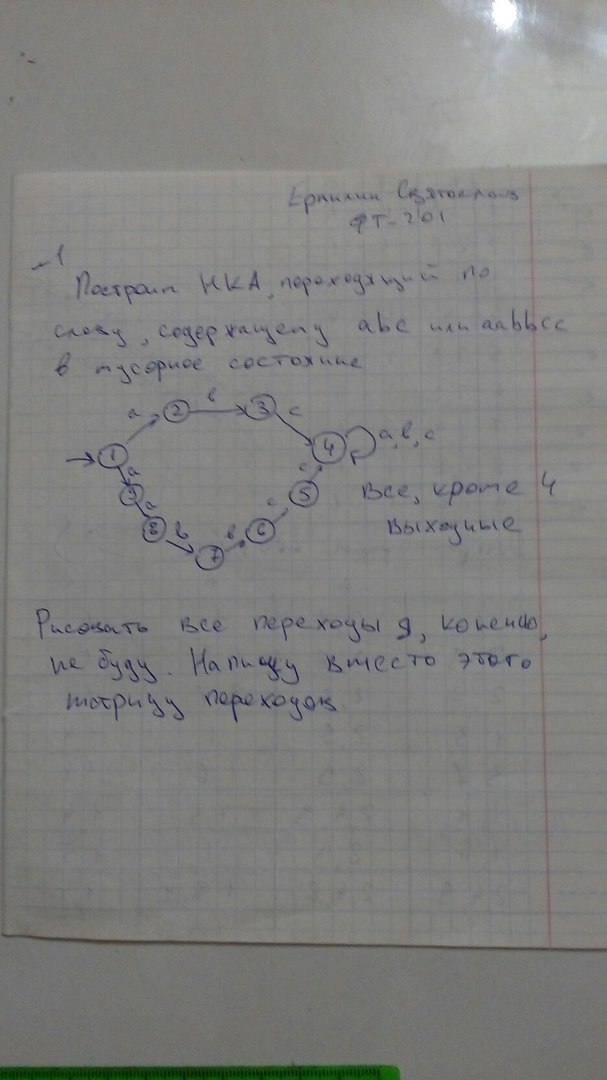
На каждой итерации берем состояние из очереди и по каждой букве алфавита запишем множество состояний, в которое мы перейдем из данного.  
Если данное состояние мы еще не выписывали - кладем его в очередь.

(Выписанным считать состояние, для которого мы проводили итерацию.)  
Повторять пока очередь не пуста.

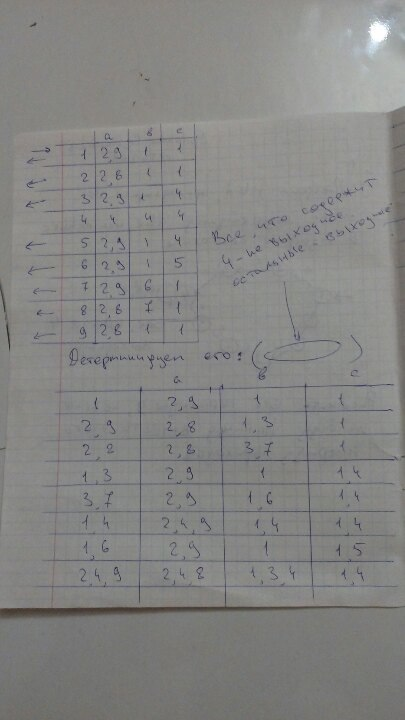
В итоге получится матрица переходов нового автомата.  
Все состояния, содержащие выходные исходного, являются выходными. Входное только самое первое, что мы выписали.

Пример:

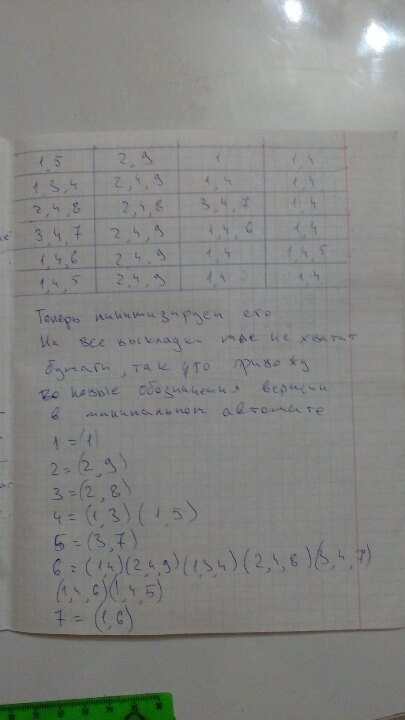
тут автомат



тут начало детерминирования



тут конец



## Минимизация автомата

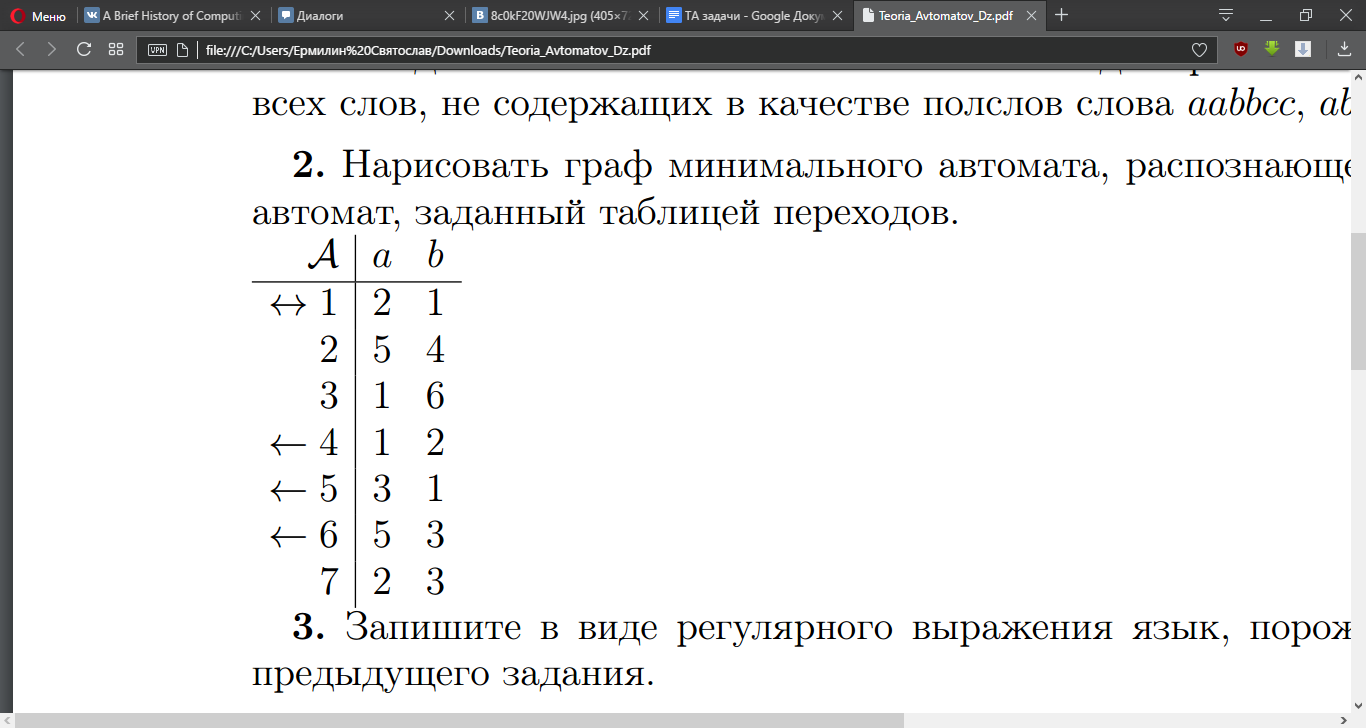
Выбросим из автомата недостижимые из входных состояния.

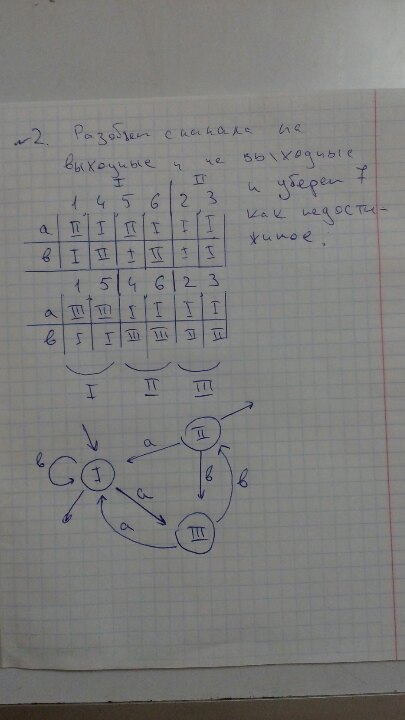
Разобьем вершины на 2 множества - выходные и не выходные.

Повторять, пока не будет получено ни одного нового множества:

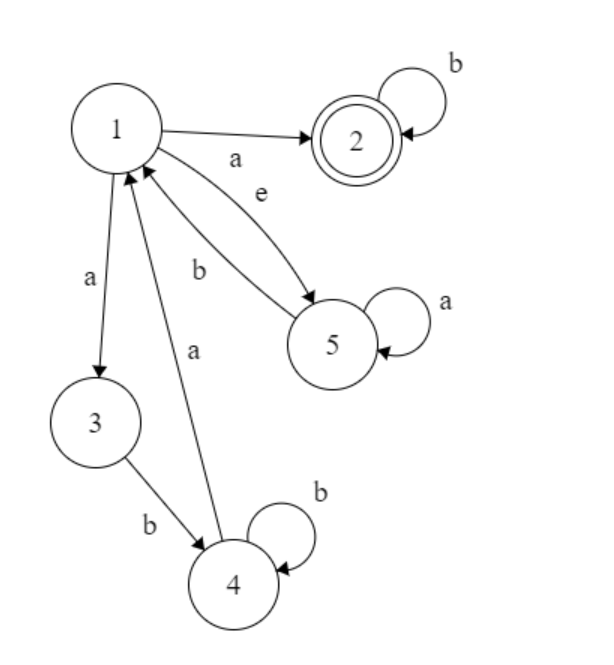
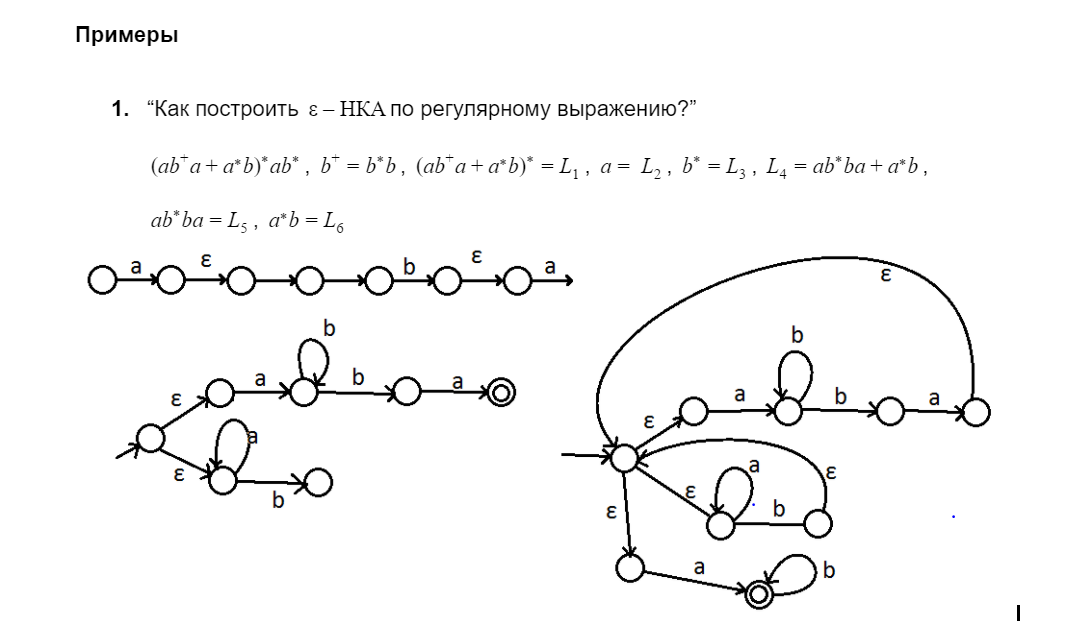
Каждому мн-ву поставим в соответствие число(для удобства).  
 Для каждого состояния выпишем, в какие множества оно переходит.  
 Состояния из одного множества с одинаковым набором переходов будем считать новым множеством.

Пример:





## Построение автомата по регвыру

- или так.

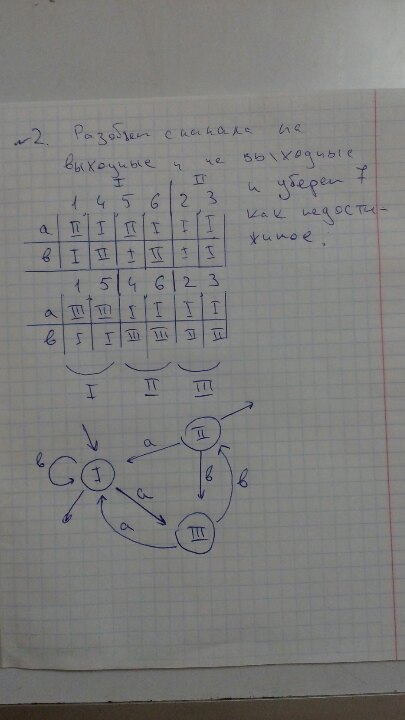
## Построение регвыра по автомату

Пусть Li - язык, распознаваемый автоматом, если i - входное.  
Для каждой вершины V: Lv = ∑sjLj, где sj - буква, по которой из V есть переход в j.  
Если состояние выходное то к его L нужно прибавить eps.  
  
Таким образом получим систему уравнений. Упростим ее подстановками.  
А далее воспользуемся тем что L = uL+v ⇔ L = u\*v

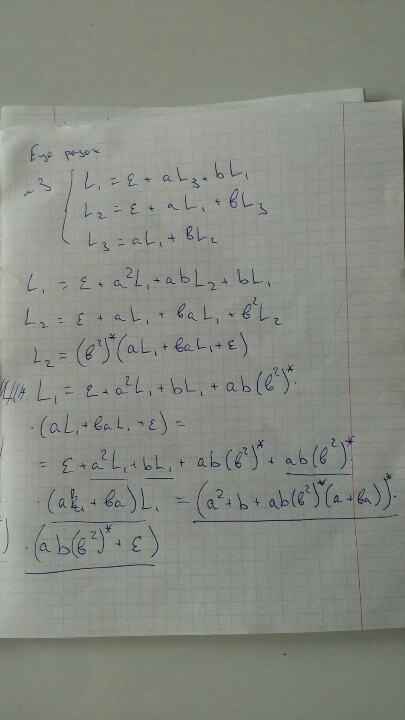
В итоге наш автомат распознает сумму L по всем входным состояниям.

#### Пример:

Автомат.



Решение.



## Лемма о накачке

### Формулировка.

Пусть — регулярный язык над алфавитом . Тогда существует такое , что для любого слова длины не меньше найдутся слова , для которых верно , , и .

### Пример.

Для доказательства нерегулярности языка часто удобно использовать отрицание леммы о разрастании. Пусть L — язык над алфавитом \Sigma. Если для любого натурального n найдётся такое слово \omega из данного языка, что его длина будет не меньше n и при любом разбиении на три слова x,y,z такие, что y непустое и длина xy не больше n, существует такое k, что xy^kz \notin L, то язык L нерегулярный.

Рассмотрим такой подход на примере языка правильных скобочных последовательностей. Для фиксированного n предъявляем слово \omega=(^n)^n. Пусть \omega как-то разбили на x, y, z. Так как |xy|\leqslant n, то y=(^b, где b > 0. Для любого такого разбиения берём k=2 и получаем xy^kz=(^{n+b})^n, что не является правильной скобочной последовательностью. Значит, язык правильных скобочных последовательностей нерегулярен.

## Моноиды всякие (кто-то умный - допишите, пожалуйста)

Будем выписывать таблицу, в столбцах которой состояния автомата, а в строках - всевозможные слова получаемые из алфавита в лексикографическом порядке.  
В пересечении - состояние в которое мы перейдем из данного по данному слову.  
Выписываем таблицу, пока не поймем, что уникальных строк больше не встретим.**(Вот тут не очень понятно как сформулировать)**(Можно выписать n^n уникальных строк, где n- состояний в автомате. И пользуясь полученными правилами уберем лишние)

После этого строим новую таблицу с уникальными строками из предыдущей в качестве строк и столбцов. В пересечении пишем слово, эквивалентное то\_что\_в\_строке+то\_что\_в\_столбце. Оно должно быть из того же множества строк.

Я не очень вижу смысл в этих несвязных попытках что-то объяснить.  
Кажется, в [конспекте](https://docs.google.com/document/d/1lGsMGmb7kHXetWKx0cu28uIkJjI6eYfRV-18Y54fxrw/edit#heading=h.nhi98xahc4fx) всё отлично написано.

Про условие, что уникальных слов больше не встретим:

Вот мы завели очередь.

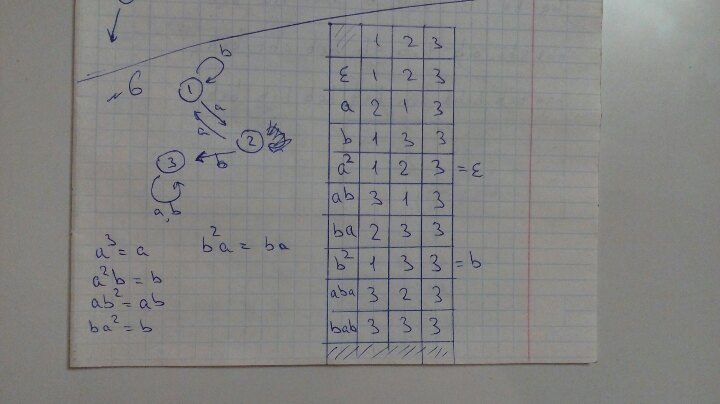
Изначально там только .

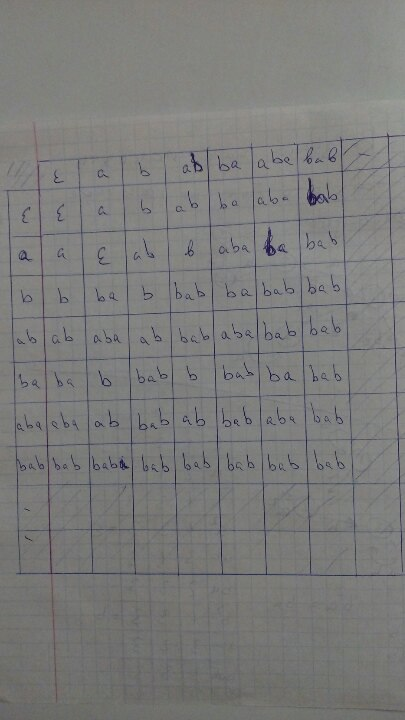
Потом мы на очередной итерации достаём из очереди строку .

Получаем строки - то есть, дописываем к строке все возможные буквы.

После этого смотрим не совпала ли какая-то из приписанных строк с уже полученными ранее. Если совпала - не складываем новую строку в очередь. Иначе - складываем.  
Так делаем, пока очередь не опустеет.

#### Пример:





## Трансдьюсеры

## Синхронизирующиеся