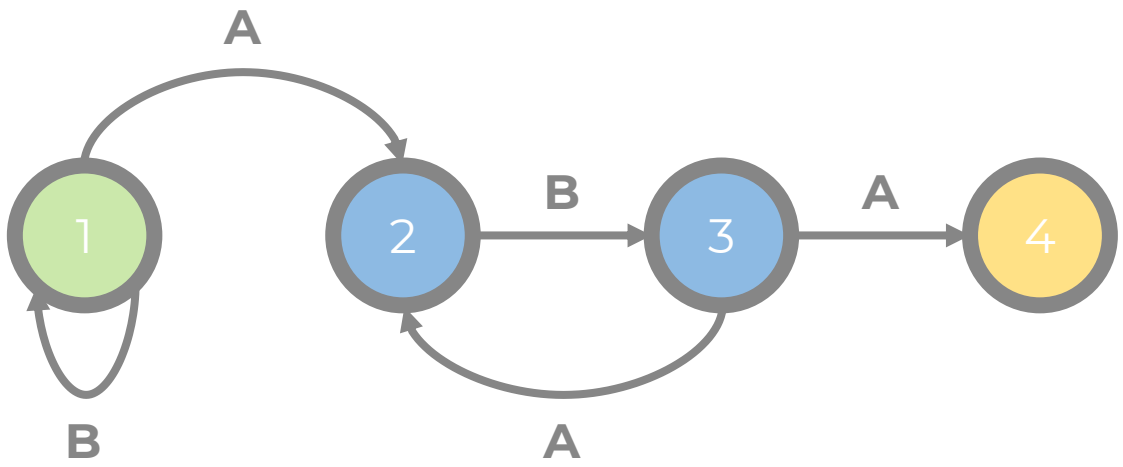


Ищем паттерны в строках по-новому!

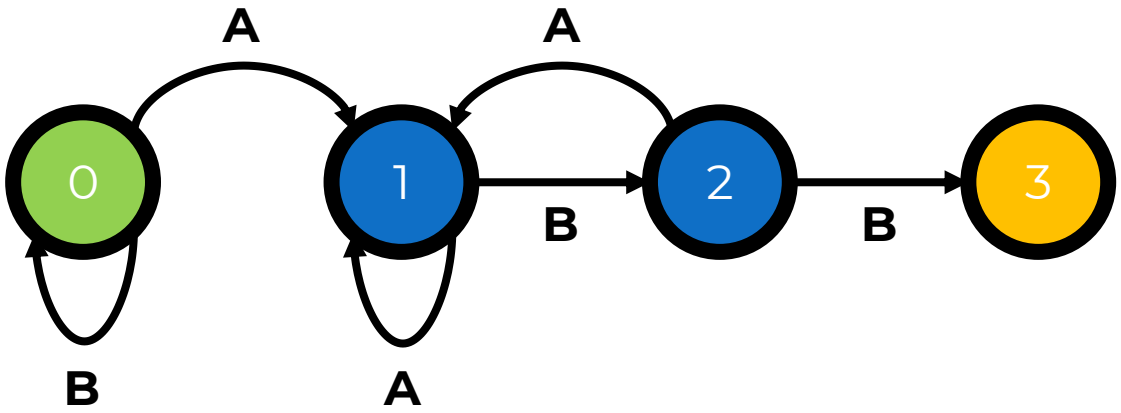
Конечные автоматы

Что у нас сегодня?

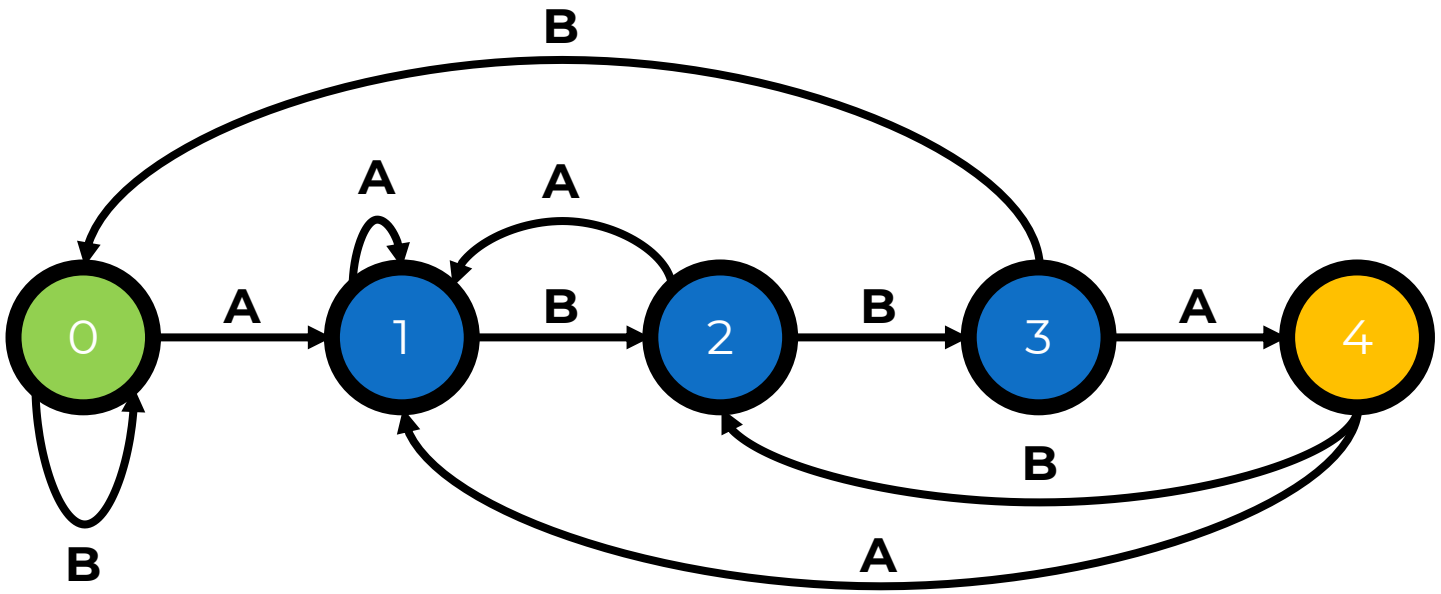
1. Конечный автомат
2. Недетерминированный автомат
3. Нагруженное дерево
4. Суффиксные ссылки
5. Хорошие суффиксные ссылки
6. Алгоритм Ахо – Корасик
7*. Построение рег. выражения

Конечный автомат

Конечный автомат – это абстрактное дискретное устройство, имеющее один вход и один выход и в каждый момент находящееся в одном состоянии из конечного набора возможных.



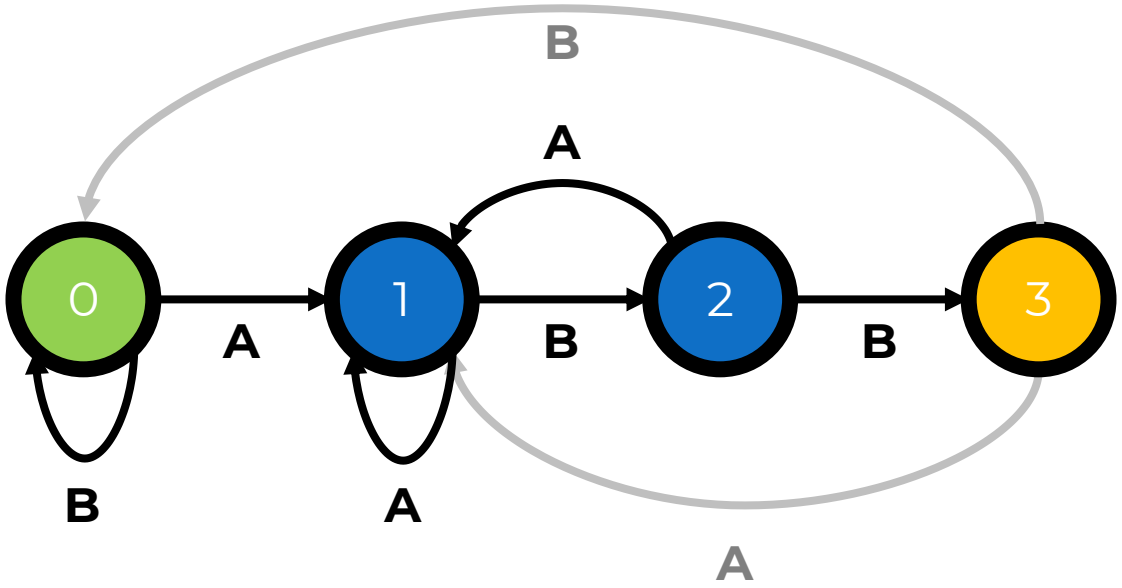
Давайте попробуем построить автомат для паттерна ABBA



Как более эффективно задать конечный автомат?

Таблицей переходов

state	A	B
0	1	0
1	1	2
2	1	3
3	1	0



За какое время мы можем вычислить таблицу переходов?

O(M)?

Почти: O(M|Σ|)
Почему так?

Можно свести построение конечного автомата к КМП

Сколько мы ищем автоматом паттерн в тексте длины N?

O(N)

Какова асимптотика поиска паттернов в тексте конечным автоматом?

В общем случае?

O(M|Σ|) + O(N)

В лучшем случае?

O(M|Σ|) + O(N)

В худшем случае?

O(M|Σ|) + O(N)

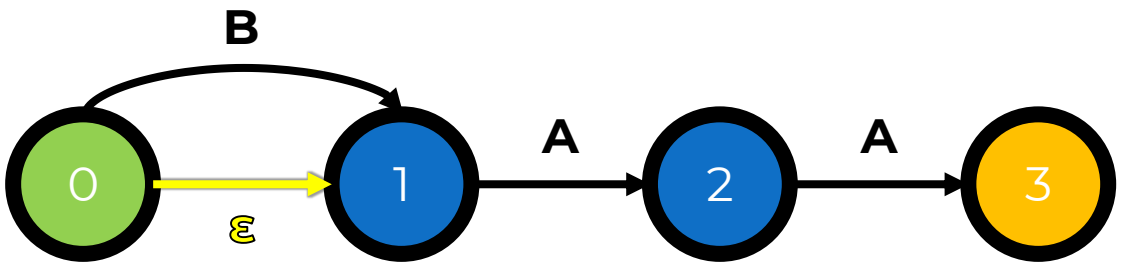
Детерминированность конечного автомата

Конечные автоматы подразделяются на **детерминированные** и **недетерминированные**.

Детерминированным называется такой конечный автомат, в котором из любого состояния по любому символу возможен переход не более чем в одно состояние.

Недетерминированный автомат

Недетерминированный автомат может содержать эпсилон-переходы (переход по пустой строке) и может переходить в несколько разных состояний по одним входным данным.

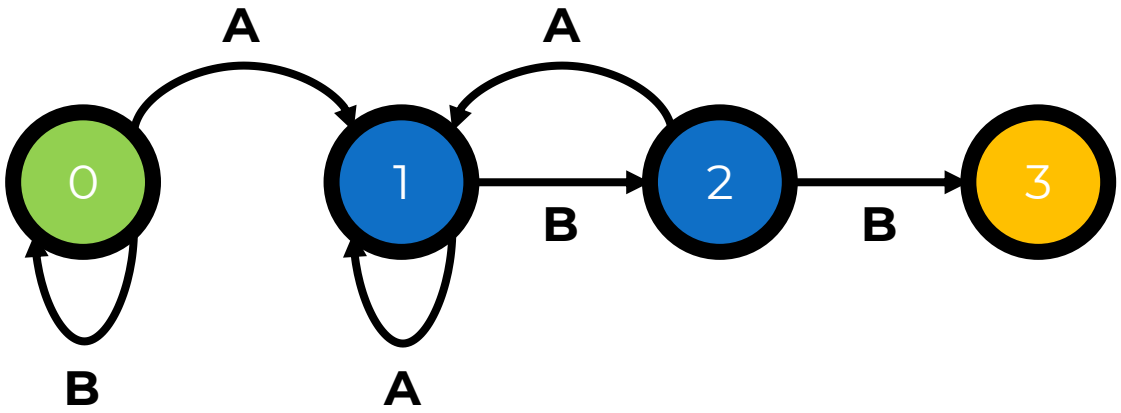


Для чего можно использовать недетерминированные автоматы?

Для построения регулярных выражений

Поиск множества паттернов в тексте

Если у нас есть **k** паттернов, максимальная длина которых **M**, за какое время можно осуществить их поиск в тексте длины **N** известными алгоритмами?



Поиск множества паттернов в тексте

Если у нас есть **k** паттернов, максимальная длина которых **M**, за какое время можно осуществить их поиск в тексте длины **N** известными алгоритмами?

Можно использовать КМП

Можно ли быстрее?

Для всех паттернов выйдет O(kM + kN)

Нагруженное дерево (бор)

А что такое бор?

Разберемся на примере...

HE

SHE

HIS

HER

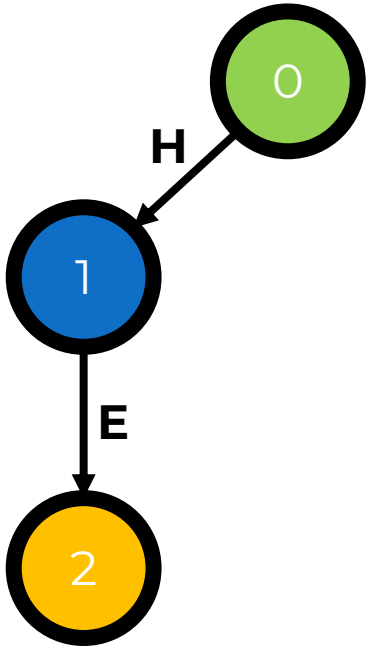
Добавим нулевую вершину – корень бора

HE

SHE

HIS

HER



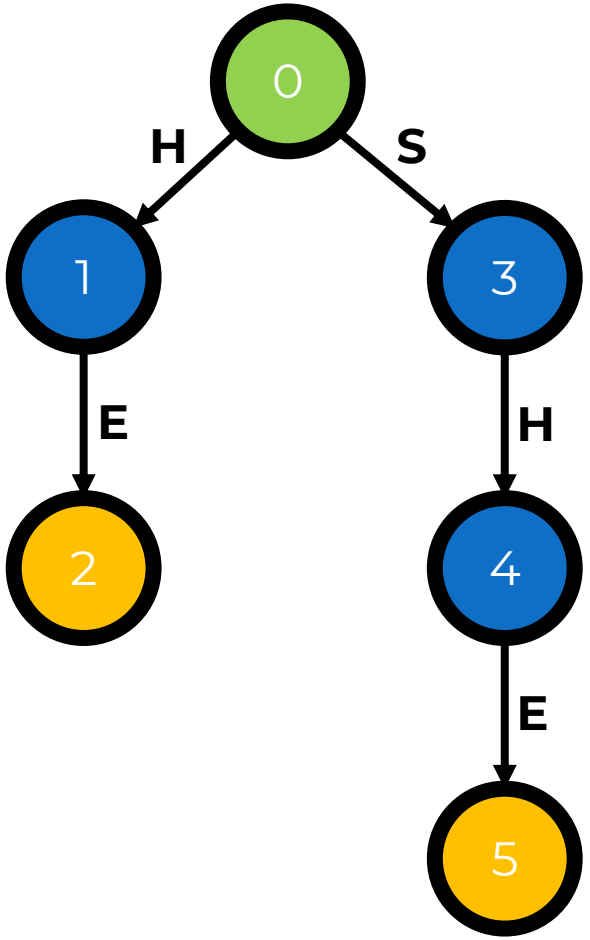
Построим состояния и переходы по первому паттерну так же, как для конечного автомата

HE

SHE

HIS

HER



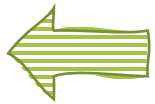
Аналогично делаем для всех остальных паттернов

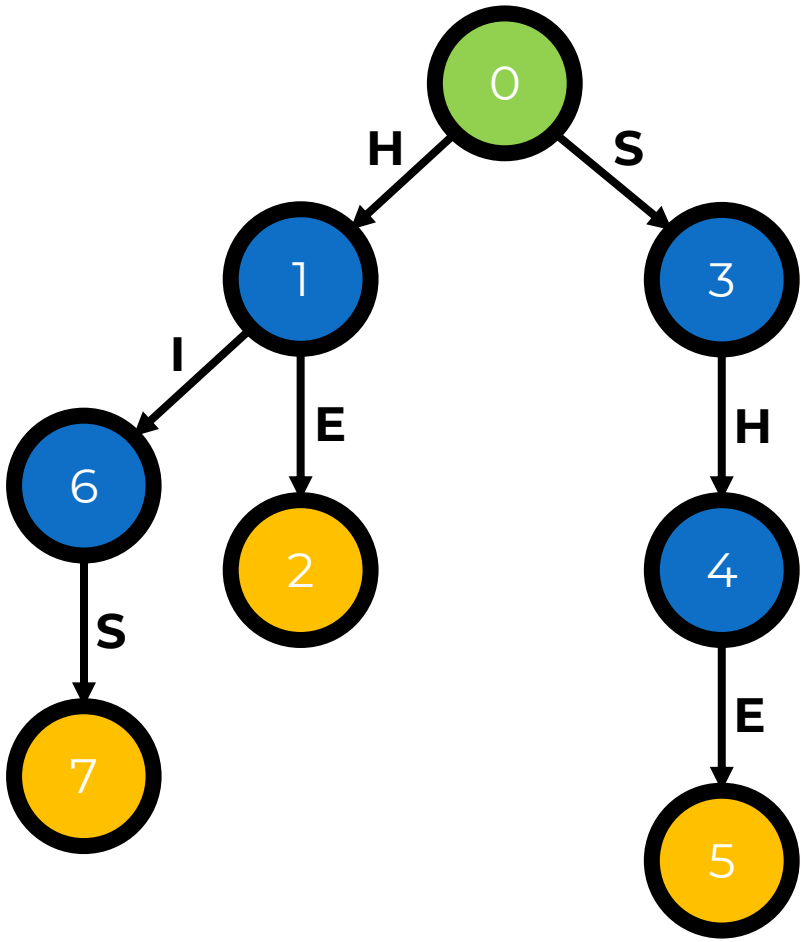
HE

SHE

HIS

HER





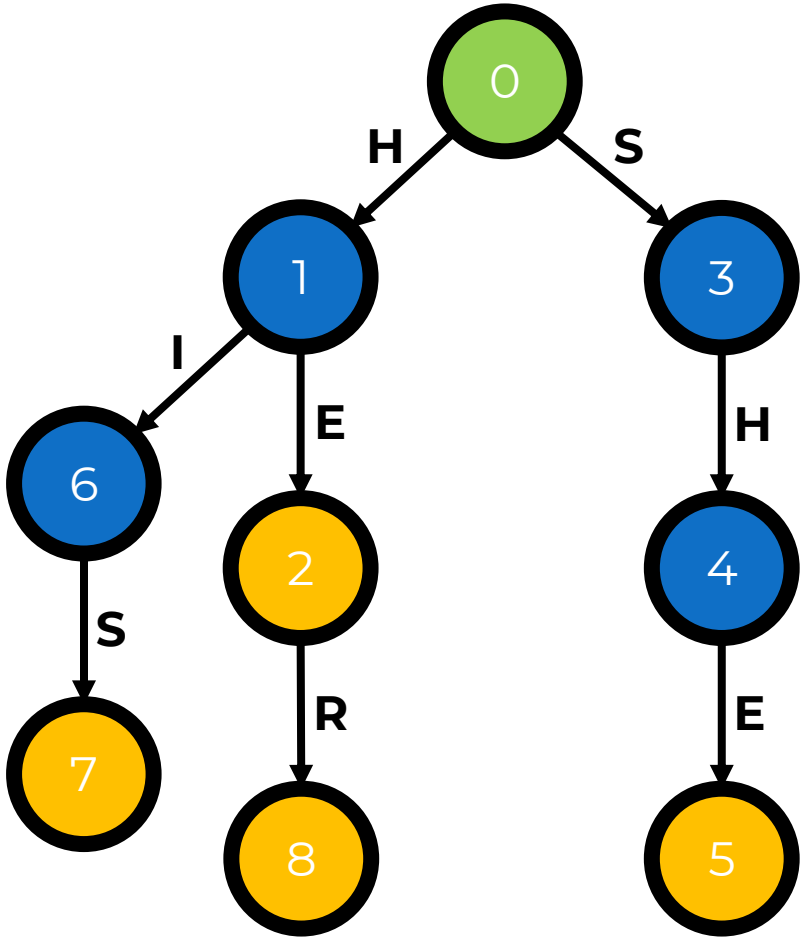
HE

SHE

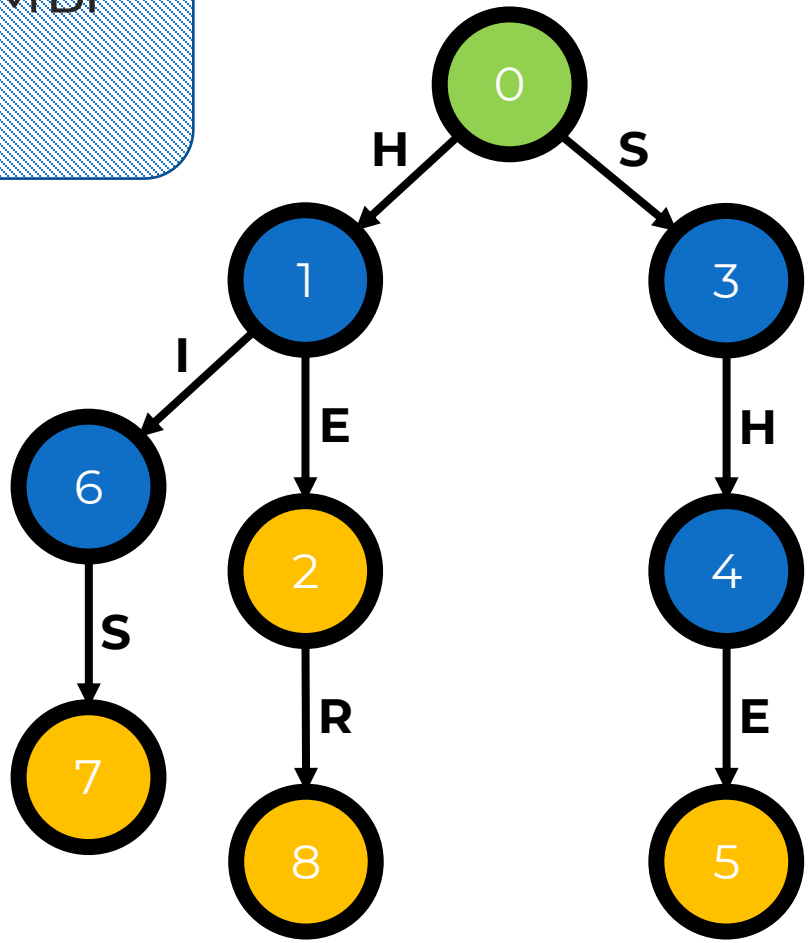
HIS

HER

Ура! Бор построен!



За какое время мы строим бор?

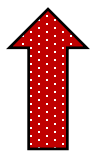


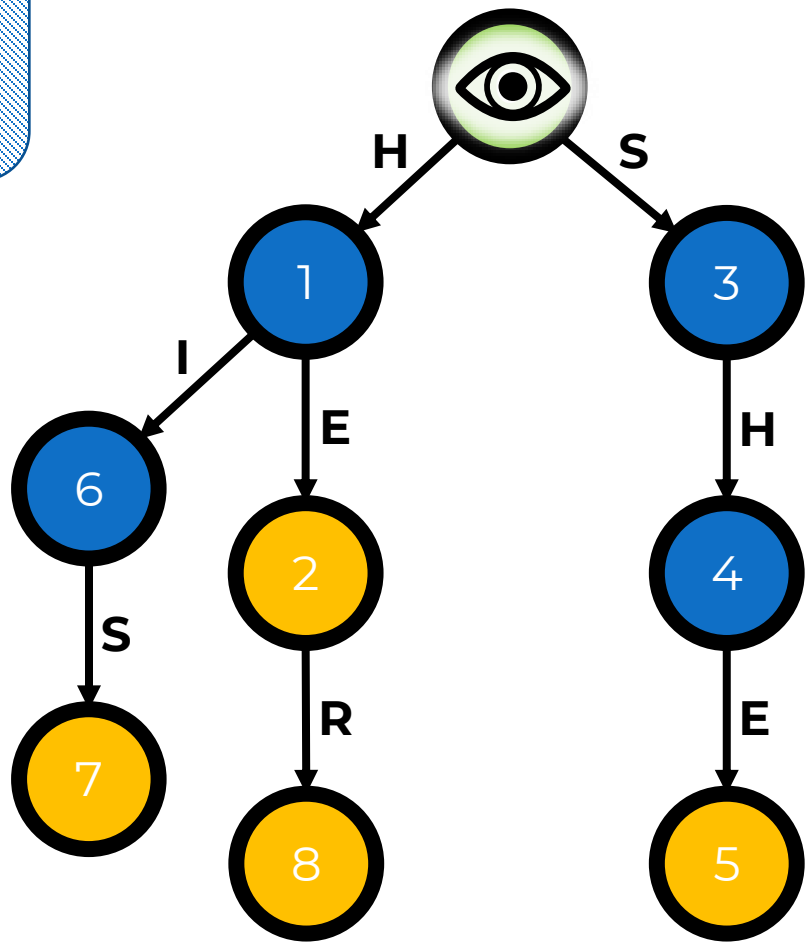
За O(kM)

Каждый паттерн добавляется за линейное время

Поищем выбранные паттерны в тексте:

HERHISHE





HE

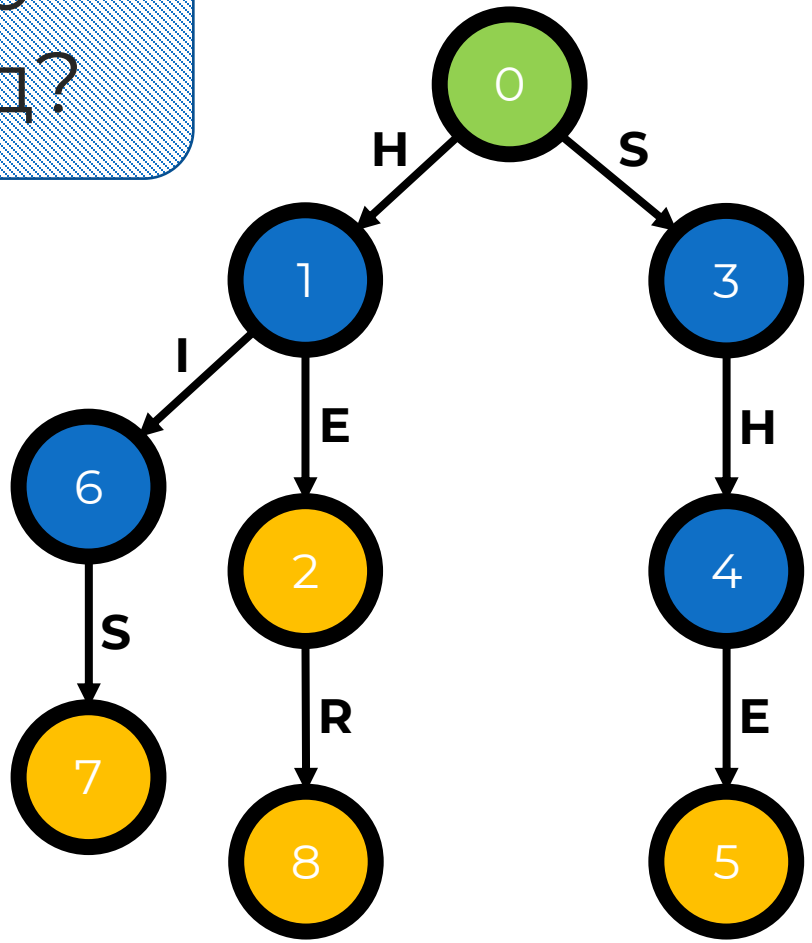
HER

HIS

HE

SHE

Какой можно сделать вывод?



Как это делать?

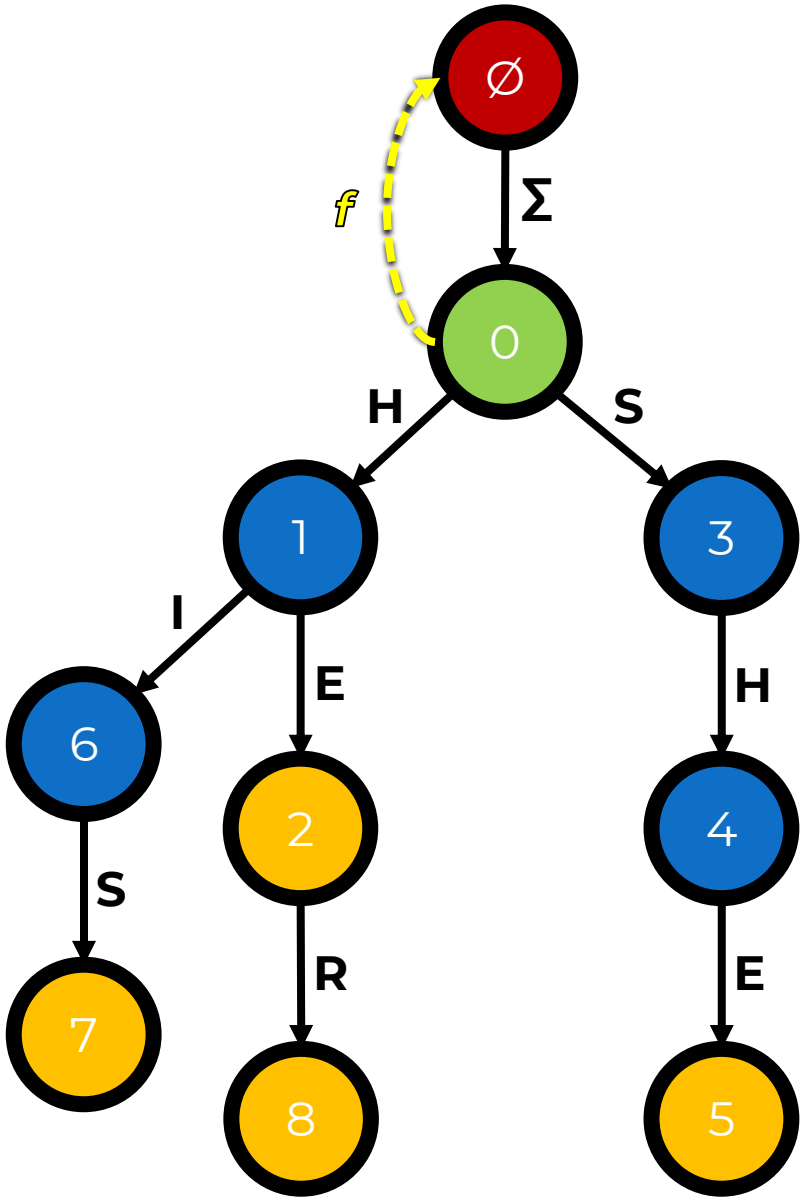
При ошибке нужно идти не в корень!

Суффиксные ссылки

Суффиксная ссылка (суффикс-ссылка, fail function) – это ссылка на состояние, соответствующее наибольшему собственному суффиксу строки, для которого в бору определена вершина (т.е. этот суффикс можно прочитать, идя от корня).

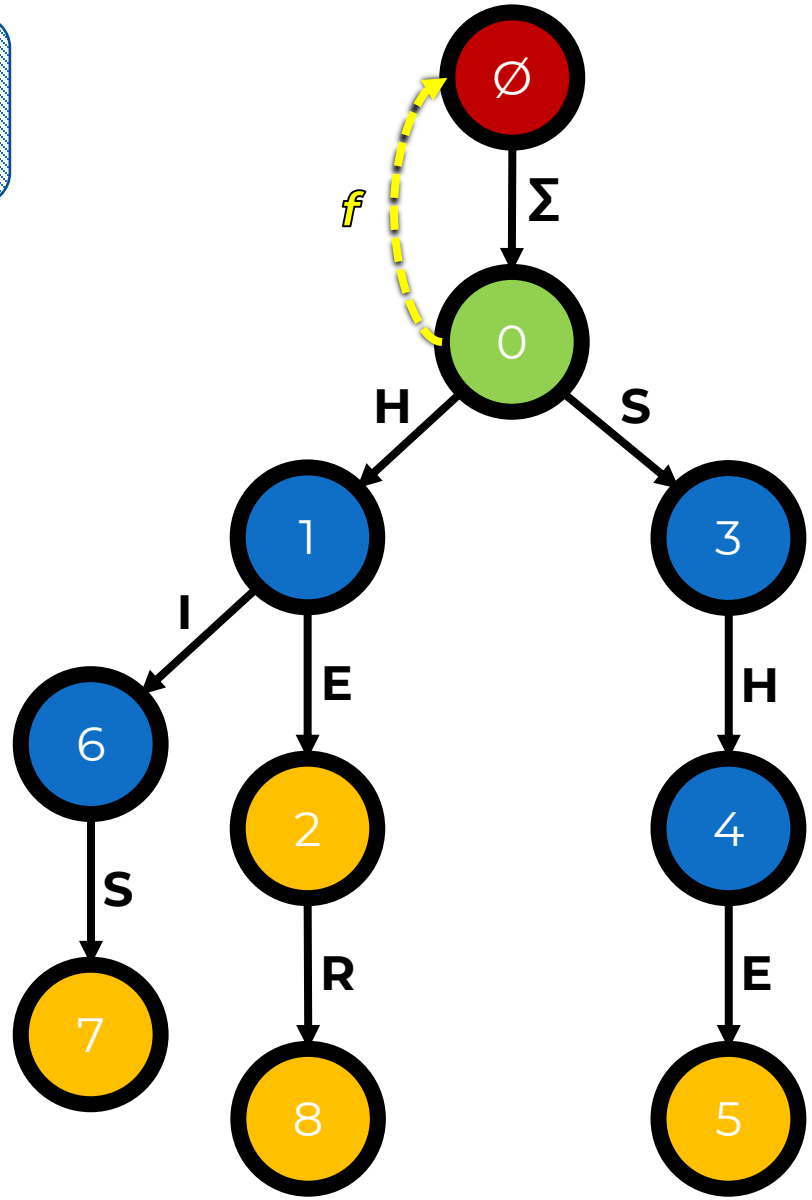
Добавим в бор особую вершину

Пусть суффикс-ссылка корня указывает на нее



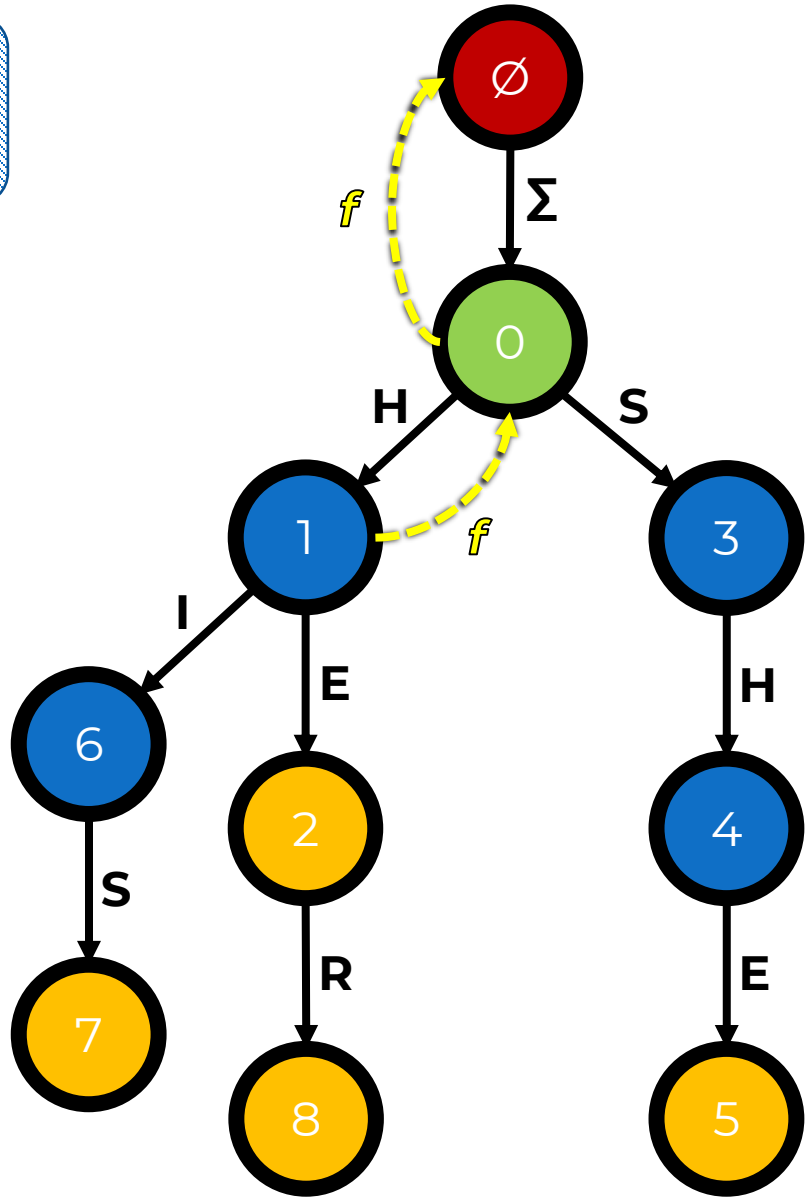
Как искать *f*-ссылки?

1. Перейдем из текущей вершины ***v*** в предка ***u***. Запоминаем символ перехода ***X***.
2. Перейдем по *f* вершины-предка ***u*** в суффикс s_1.
3. Попытаемся перейти из вершины s_i по символу ***X*** в вершину ***t***.
   a. Если переход возможен, ставим *f* из вершины ***v*** в вершину ***t***.
   b. Если перейти нельзя, то переходим по *f* вершины s_i в s_{i+1} и повторяем с шага 3.



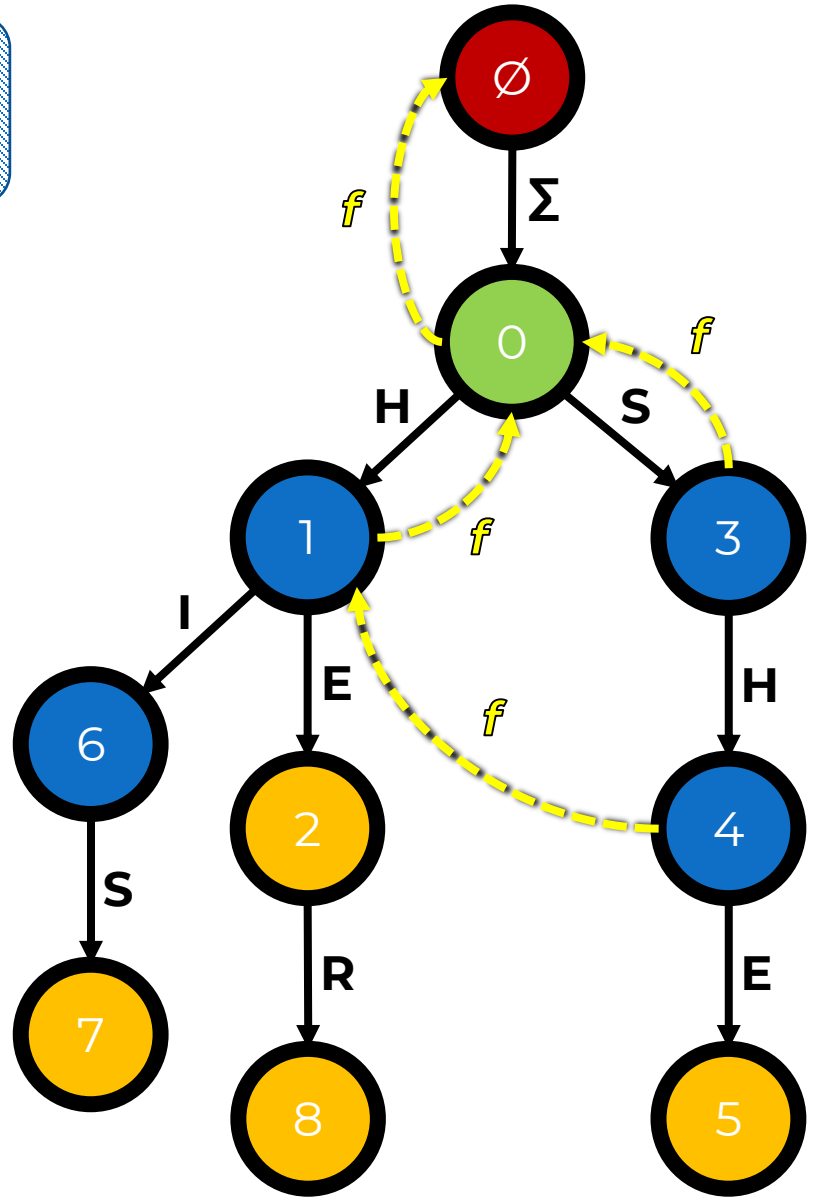
Найдем *f* для вершины 1:

1. Перейдем в предка (0) текущей вершины (1). Запоминаем символ (H).
2. Перейдем по *f* вершины-предка (0) в суффикс (Ø).
3. Попытаемся перейти из этой вершины (Ø) по символу (H) в (0).
   a. Переход возможен, ставим *f* из вершины (1) в вершину (0).



Найдем *f* для вершины 4:

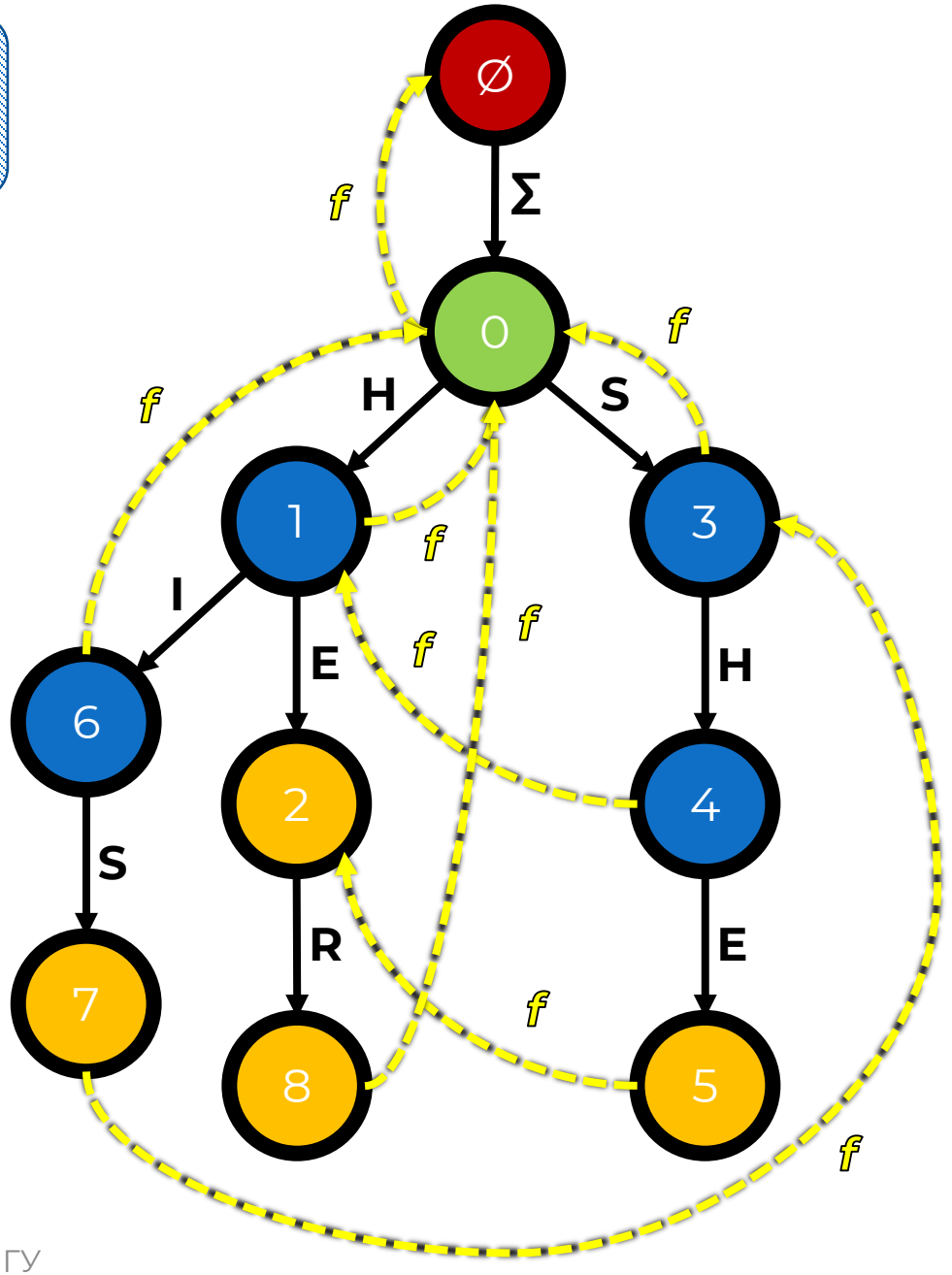
1. Перейдем в предка (3) текущей вершины (4). Запоминаем символ (H).
2. Перейдем по *f* вершины-предка (3) в суффикс (0).
3. Попытаемся перейти из этой вершины (0) по символу (H) в (1).
   a. Переход возможен, ставим *f* из вершины (4) в вершину (1).



Найдем все остальные *f* :

Чтобы избежать нагромождения, не будем подписывать *f*

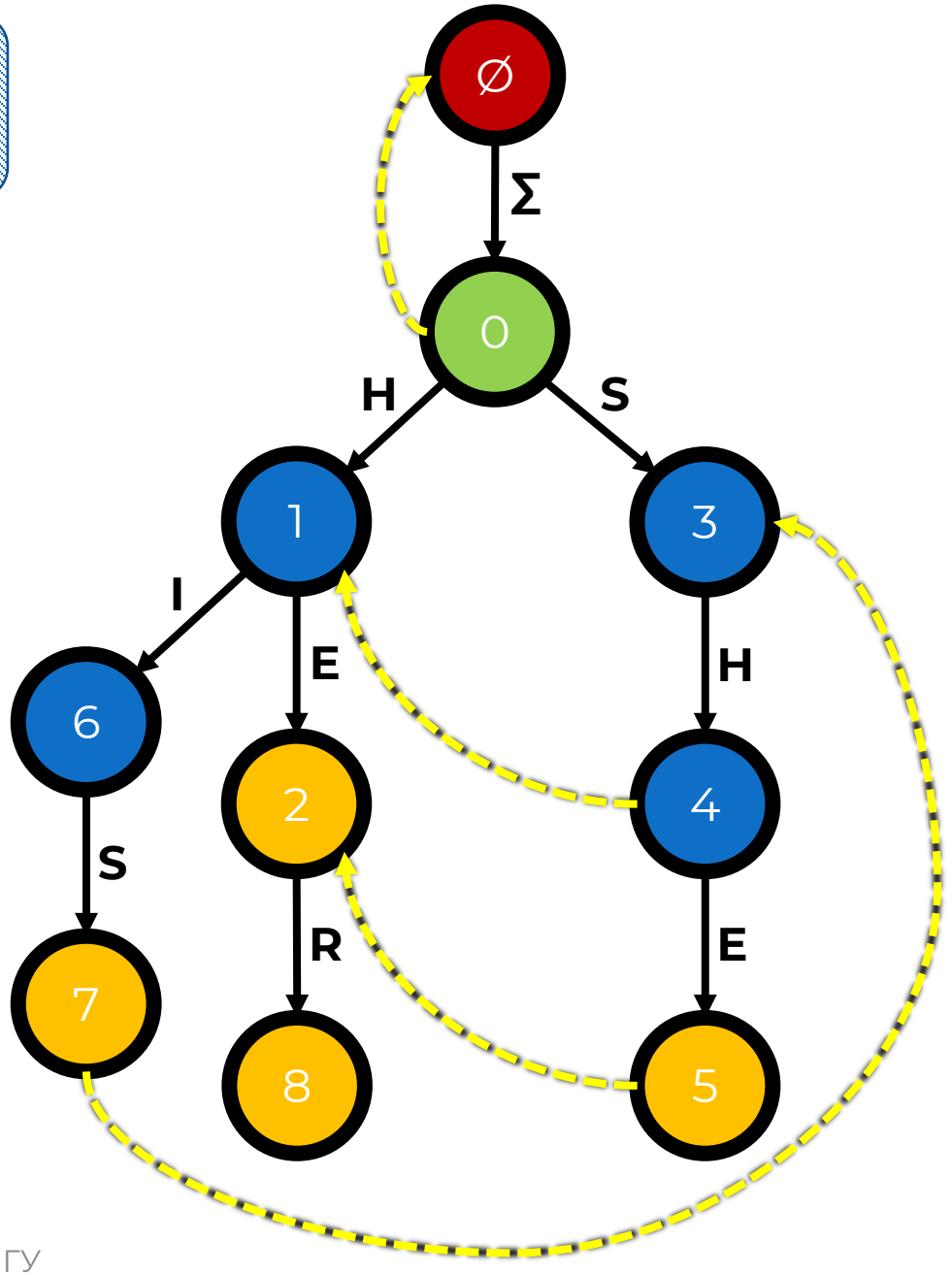
И уберем *f*, ведущие в корень



Найдем все остальные *f* :

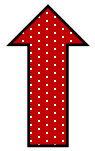
Чтобы избежать нагромождения, не будем подписывать *f*

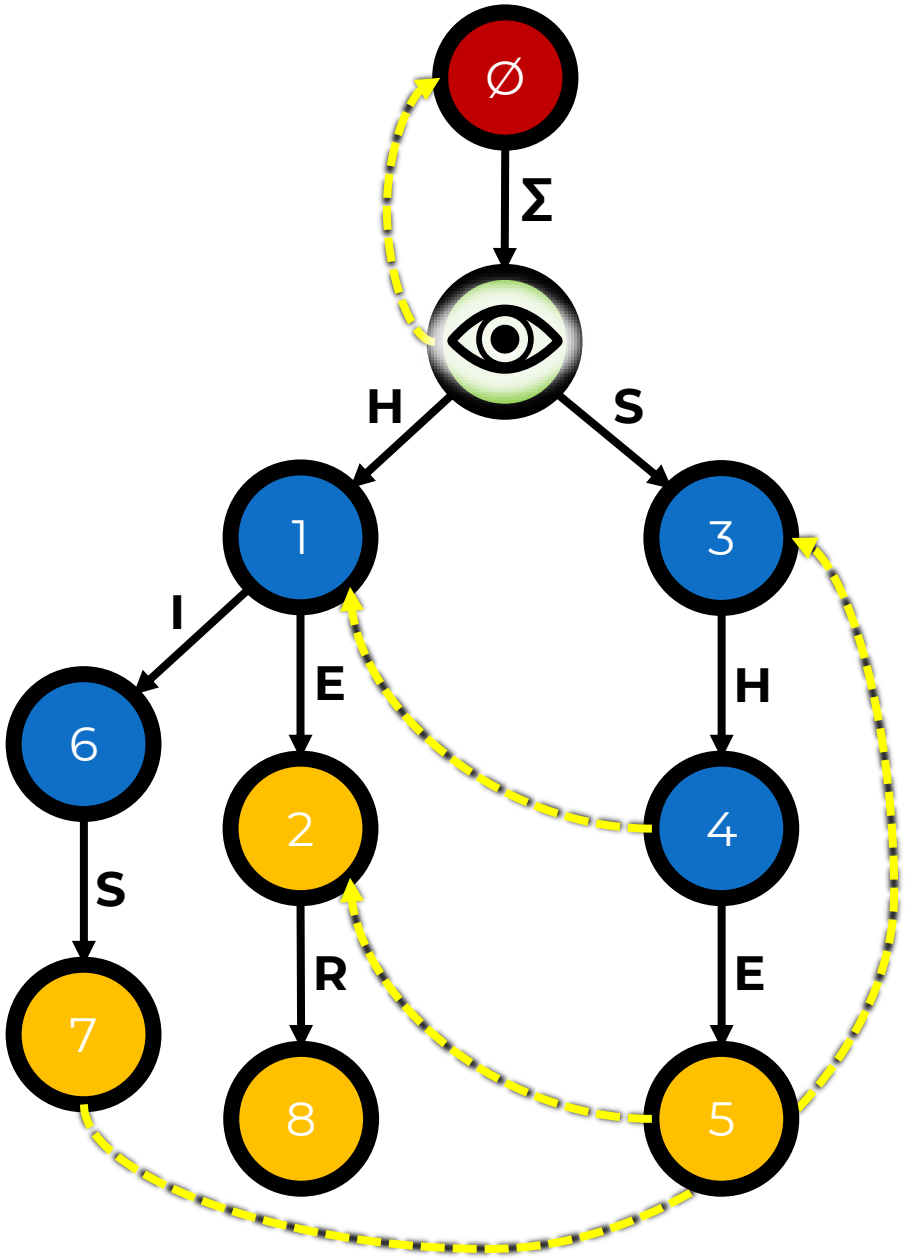
И уберем *f*, ведущие в корень



Поищем выбранные паттерны в тексте:

HERHISHE





HE
HER
HIS
SHE
HE

Время построения суффикс-ссылок

За какое время можно построить *f*-ссылки?

f-ссылки вычисляются для каждой вершины ровно один раз.

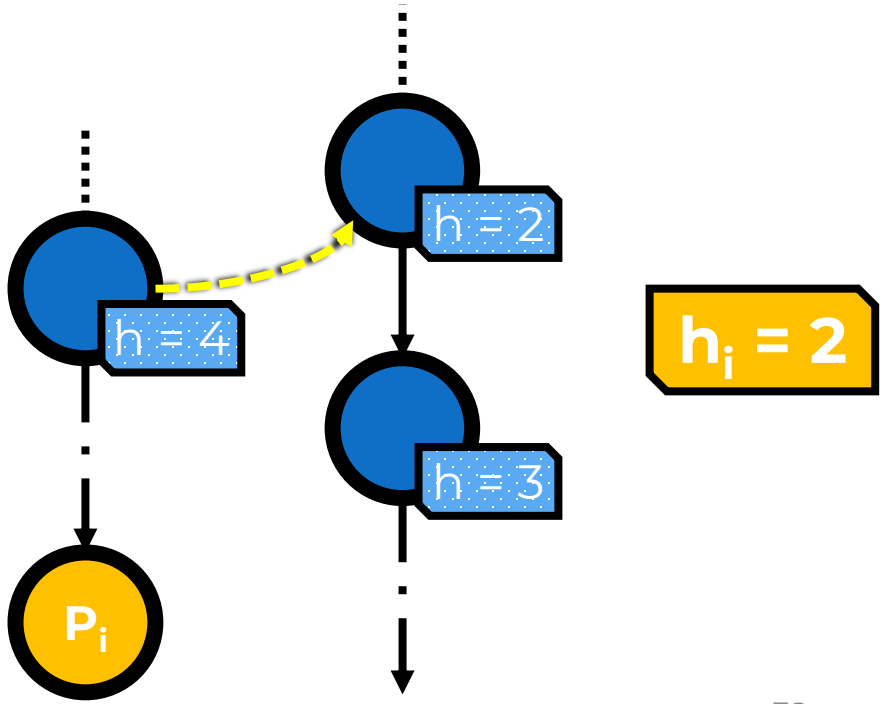
Время тратится на создание самих *f*-ссылок и переходы в вершины по ключам и *f*-ссылкам.

Значит, их **не больше O(kM)**

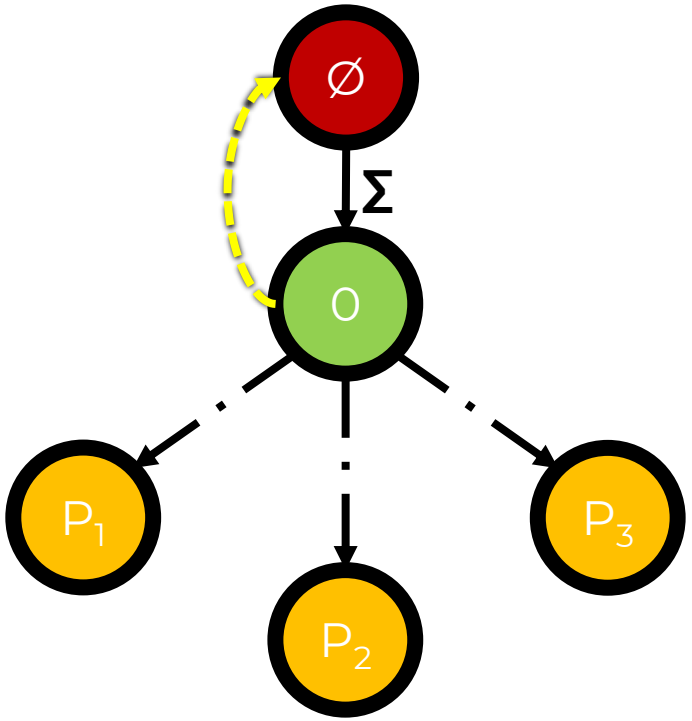
Кроме этого, время тратится на переходы по *f*-ссылкам

Используем метод потенциалов, чтобы посчитать число операций в этой процедуре.

Для каждого паттерна P_i будем хранить значение h_i, равное глубине вершины, на которую указывает **последняя** найденная *f*-ссылка в пути этого паттерна.



h корня принята за 1,
h вершины Ø – за 0.



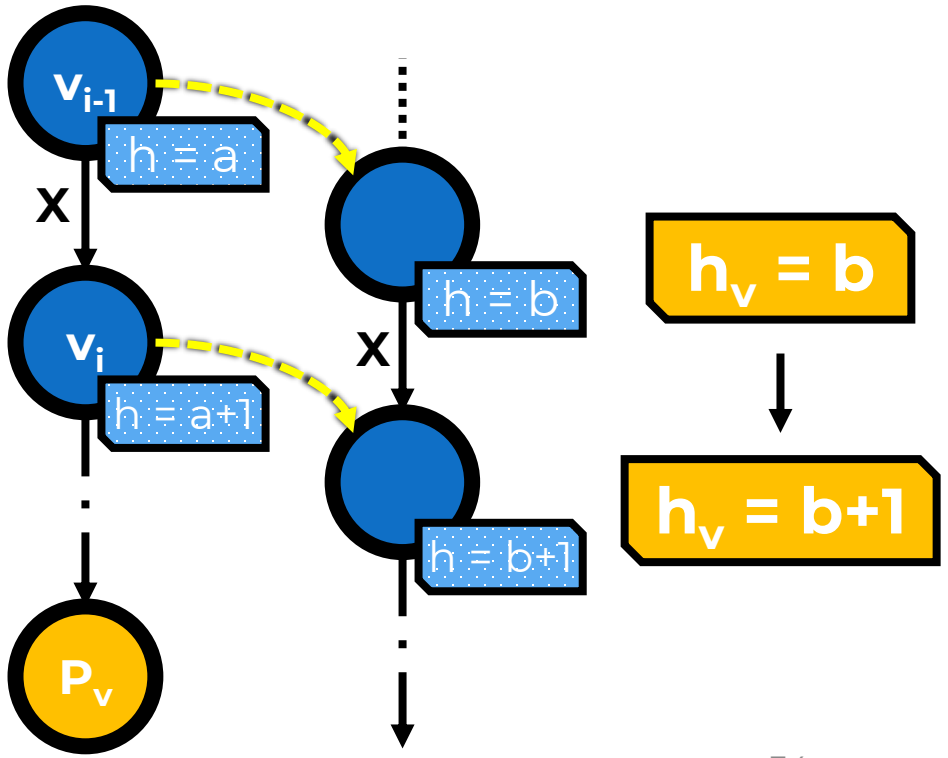
$\mathbf{h_i}$ – глубина (h) вершины, на которую указывает последняя посчитанная ***f***-ссылка в пути паттерна $\mathbf{P_i}$

Считаем **H** – сумму по $\mathbf{h_i}$. В начале работы алгоритма она равна 0.

В бору нет отрицательных высот, поэтому **H ≥ 0** на каждом шаге алгоритма

В каком случае **H** увеличивается?

Если при подсчете *f*-ссылки для вершины v_i мы совершаем *ровно один* переход по *f*-ссылке (из v_{i-1}), то h_v увеличивается на 1.



Соберем все соображения воедино:

H не может быть больше **O(kM)** в любой момент работы алгоритма.

Каждый дополнительный переход по *f* уменьшает **H** как минимум на 1

Значит, по *f*-ссылкам было сделано:

- Не более **O(kM)** переходов по *f*-ссылке узла-предка
- Не более **O(kM)** переходов по дополнительным *f*

Всего не более **O(kM)** переходов.

Всего нужно найти не более **O(kM)** *f*-ссылок

Время поиска автоматом с *f*-ссылками

Что будет, если мы **всегда** будем переходить по *f*-ссылкам?

Получается недетерминированный автомат, который работает за **O(kMN)**

Фактически они становятся **ε-переходами**

Поиск паттернов поочередно с помощью КМП работает быстрее...

Нужно ходить только в окончания паттернов?

Почти, но это тоже будет работать не всегда

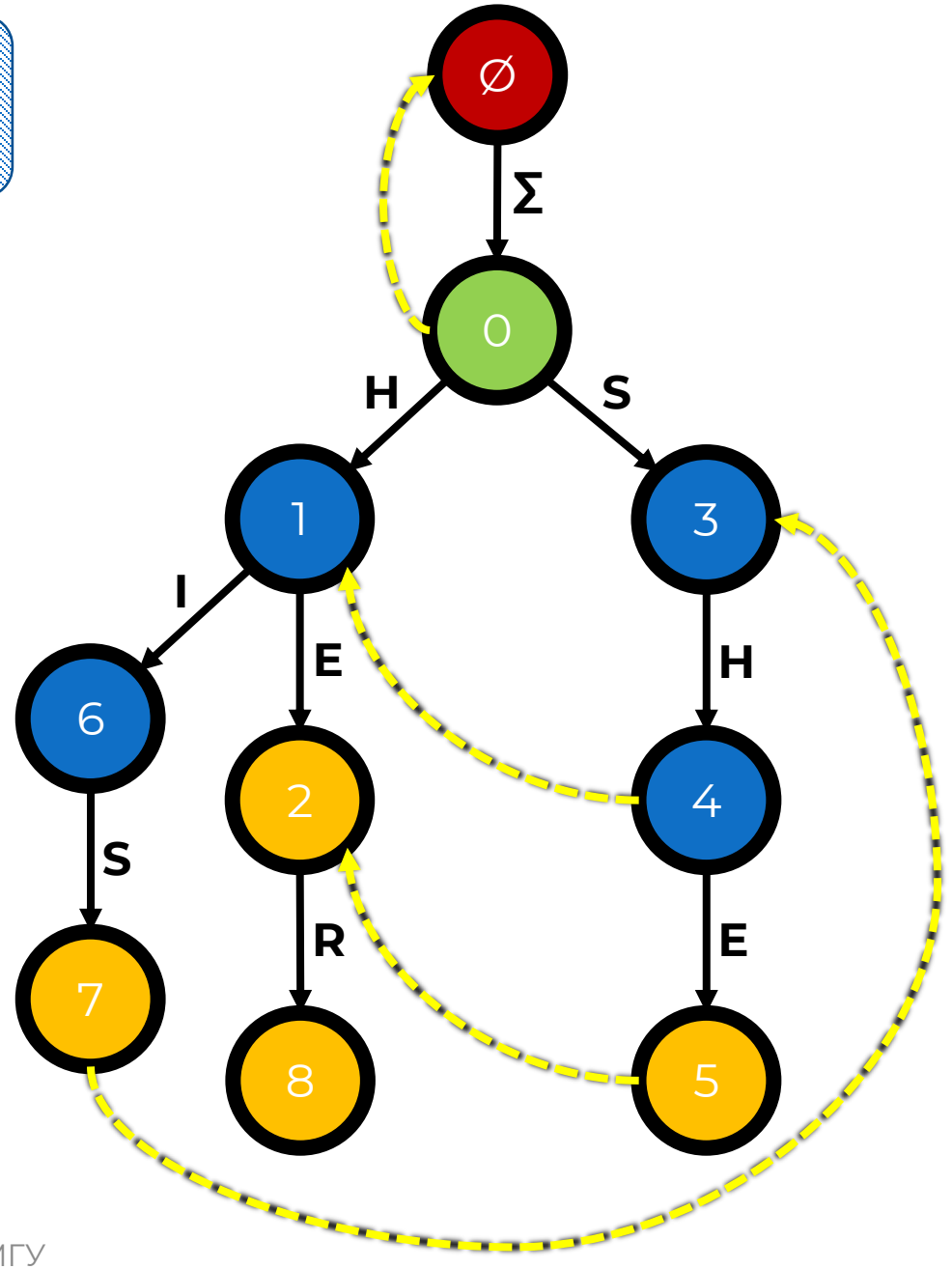
Хорошие суффикс-ссылки

Хорошей суффиксной ссылкой (сжатой, output function) вершины называют ссылку, ведущую в наибольший суффикс префикса паттерна, соответствующего этой вершине, который также является паттерном.

Если такого суффикса нет, то хорошая суффиксная ссылка **ведет в корень**.

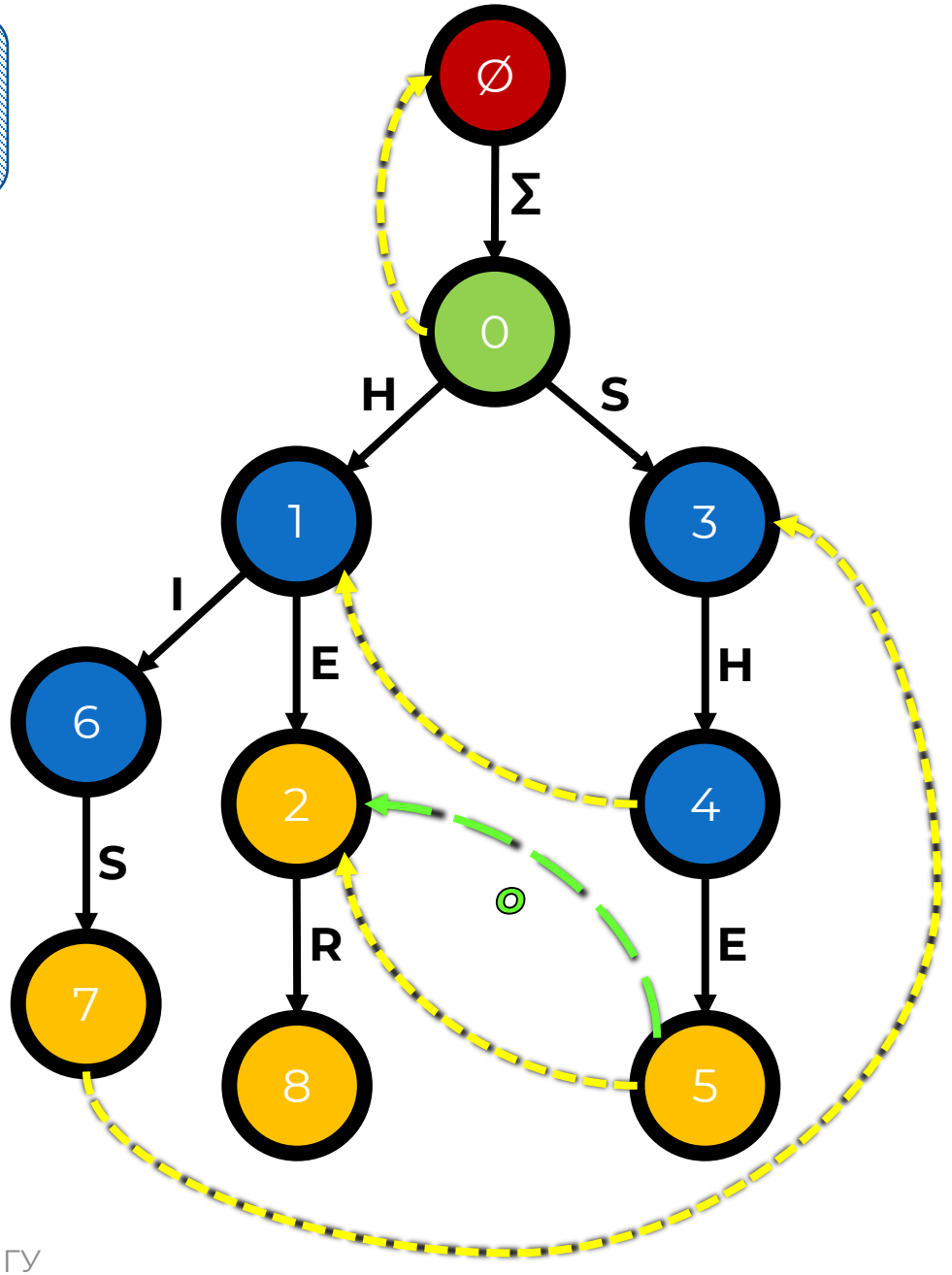
Как искать хорошие суффикс-ссылки (o)?

1. Перейдем по *f* текущей вершины s_0 в суффикс s_1.
2. Если s_i является концом паттерна или корнем, то o вершины s_0 ведет в s_i.
3. В противном случае перейдем по *f* вершины s_i в вершину s_{i+1} и повторяем с шага 2.



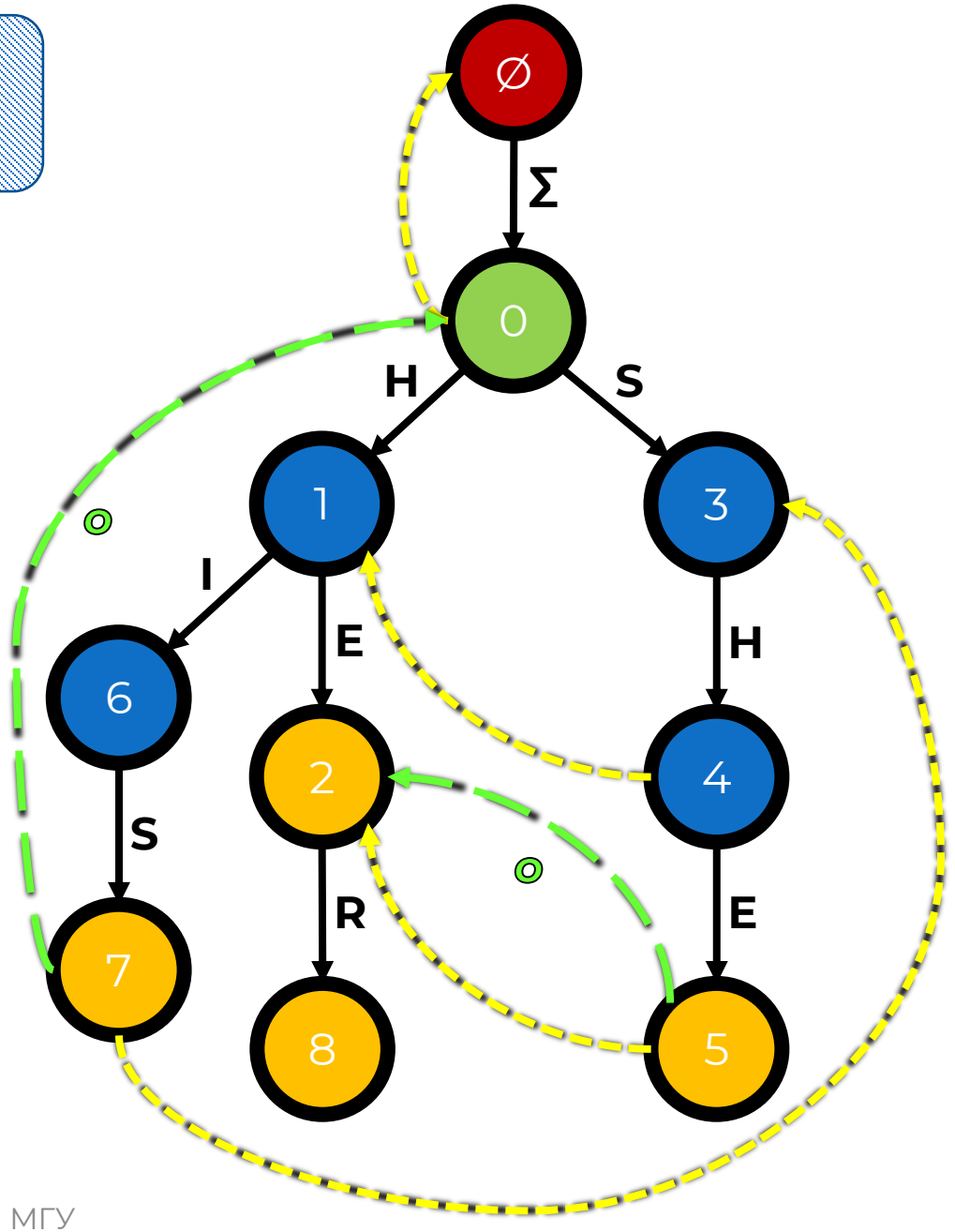
Найдем o для вершины 5:

1. Перейдем по *f* текущей вершины (5) в суффикс (2).
2. (2) является концом паттерна или корнем; o вершины (5) ведет в (2).



Найдем o для вершины 7:

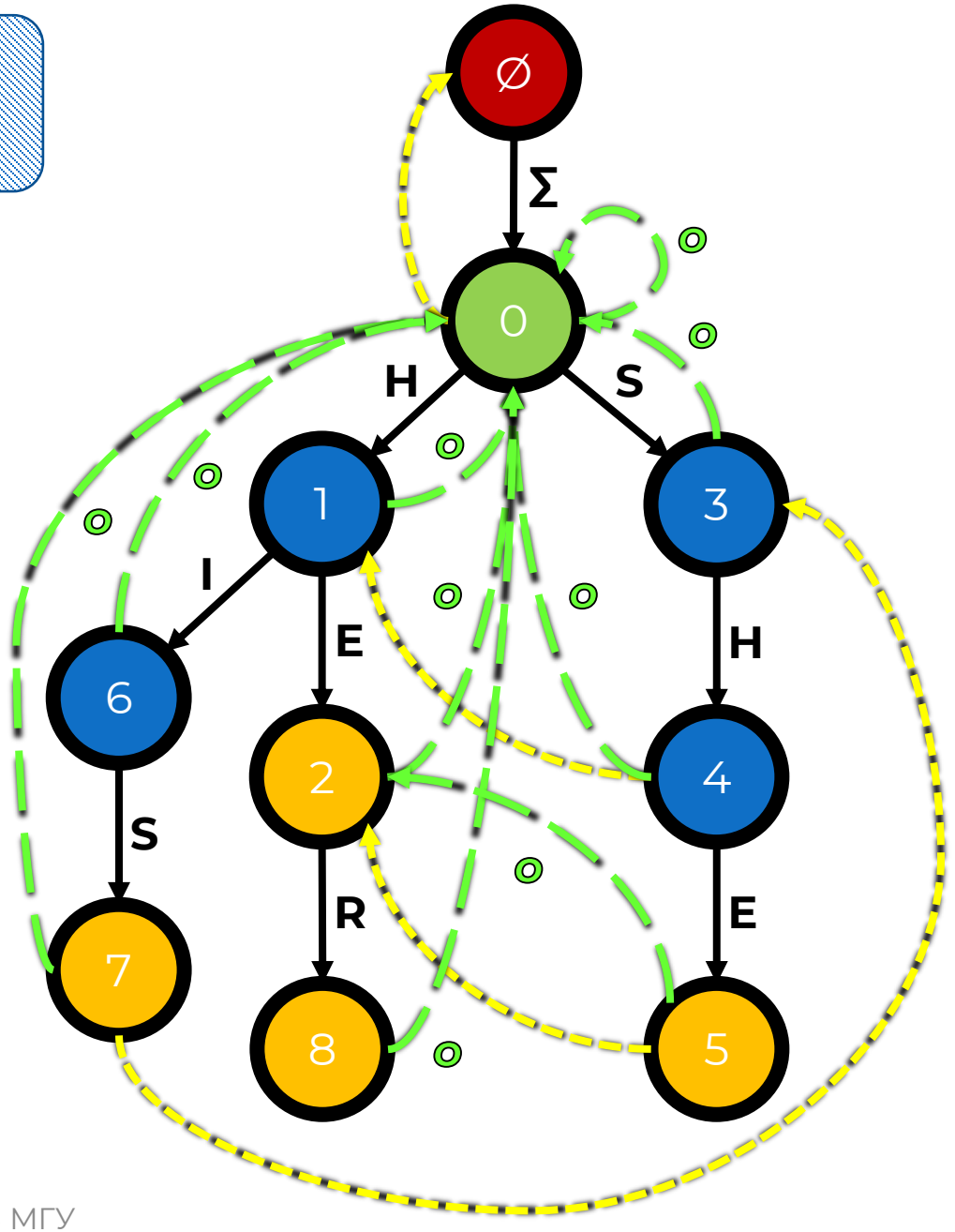
1. Перейдем по *f* текущей вершины (7) в суффикс (3).
2. (3) не является концом паттерна или корнем, переходим по *f* текущей вершины (3) в суффикс (0).
3. (0) является концом паттерна или корнем; o вершины (7) ведет в (0).



Найдем все остальные o:

Чтобы избежать нагромождения, не будем подписывать o

И уберем o, ведущие в корень



Алгоритм Ахо – Корасик

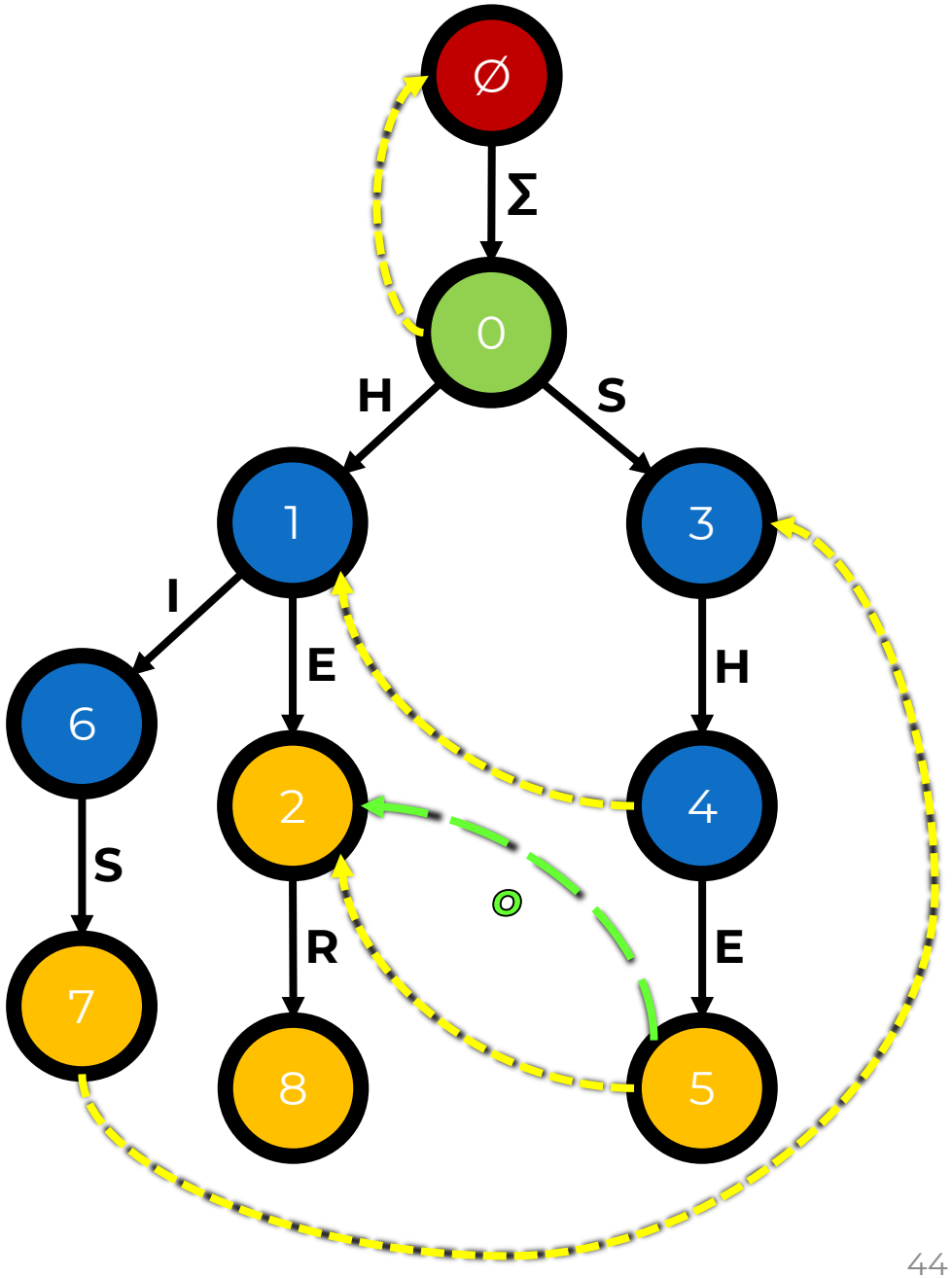
Полученный нами **конечный автомат** в виде бора с суффикс-ссылками и хорошими суффикс-ссылками был предложен Альфредом Ахо (Alfred Aho) и Маргарет Корасик (Margaret Corasick) в 1975 году.

Автомат используется в алгоритме поиска конечного набора строк в тексте (алгоритм Ахо – Корасик).

На этом алгоритме основана консольная утилита **grep**.

Как производить поиск с автоматом Ахо – Корасик?

1. Начинаем в корне.
2. Пытаемся совершить переход по следующей букве текста.
   - Если не получилось, то переходим по *f* и повторяем.
3. После перехода доходим по *o* до корня.



Каково время поиска паттернов в тексте алгоритмом Ахо – Корасик?

Оно складывается из трех составляющих:

Хождение по основным переходам

Хождение по *f*-ссылкам

Хождение по *o*-ссылкам

Сколько времени занимает хождение по основным переходам?

По одному переходу на каждую букву текста – **O(N)**

Сколько времени занимает хождение по *f*-ссылкам?

h вершины уменьшается при каждом переходе по *f*-ссылке, а увеличивается при переходе по букве текста не более чем на 1

Не более O(N) переходов

Сколько времени занимает хождение по o-ссылкам?

После каждого прохождения по основному переходу мы переходим по o-ссылке – **O(N)** раз

Кроме того, каждый дополнительный переход – это вхождение паттерна – **α** раз

Всего **O(N + α)** переходов

Сколько времени занимает поиск?

В итоге:

$O(N) + O(N) + O(N + \alpha) = \mathbf{O(N + \alpha)}$

Сколько времени занимает весь алгоритм?

$O(kM) + O(N + \alpha) = \mathbf{O(kM + N + \alpha)}$

Дополнительные темы

2 ноября 2021 г.

ФББ МГУ

Построение регулярного выражения

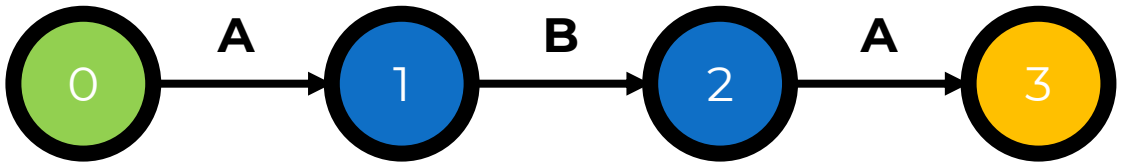
Для того чтобы построить недетерминированный конечный автомат для регулярного выражения:

`(aba)+(ba)*(bc)?b`

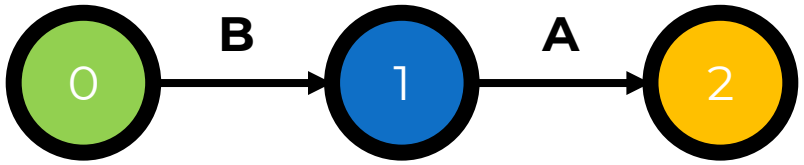
1. Построить отдельные детерминированные автоматы* для частей регулярного выражения:

*считаем, что все переходы кроме основных ведут к смерти состояния

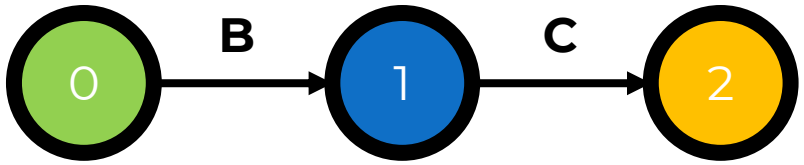
aba



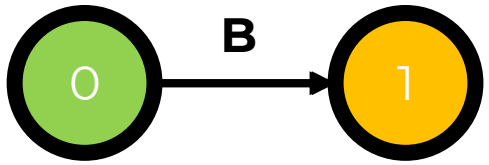
ba



bc



b



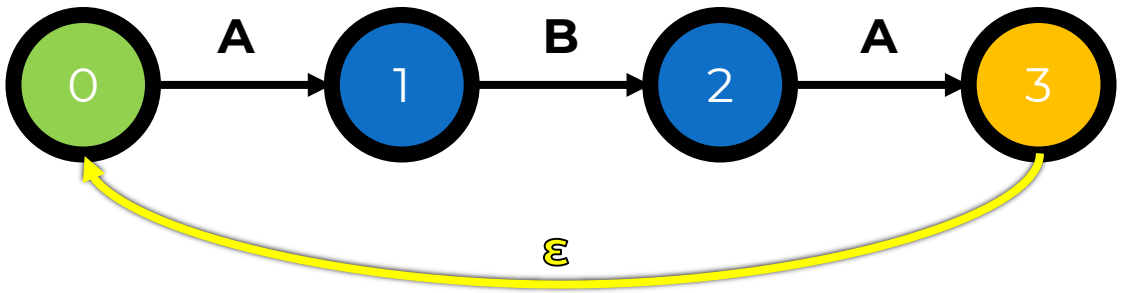
Построение регулярного выражения

2. Добавить к каждой части соответствующие модификаторы

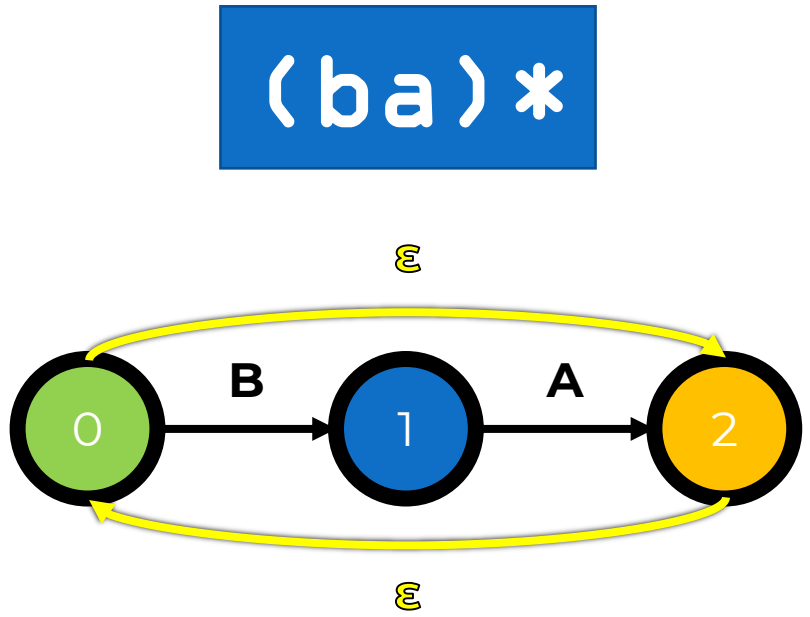
```
(aba)+(ba)*(bc)?b
```

- Для **(...)+** добавить ε-переход из конечного состояния в стартовое:

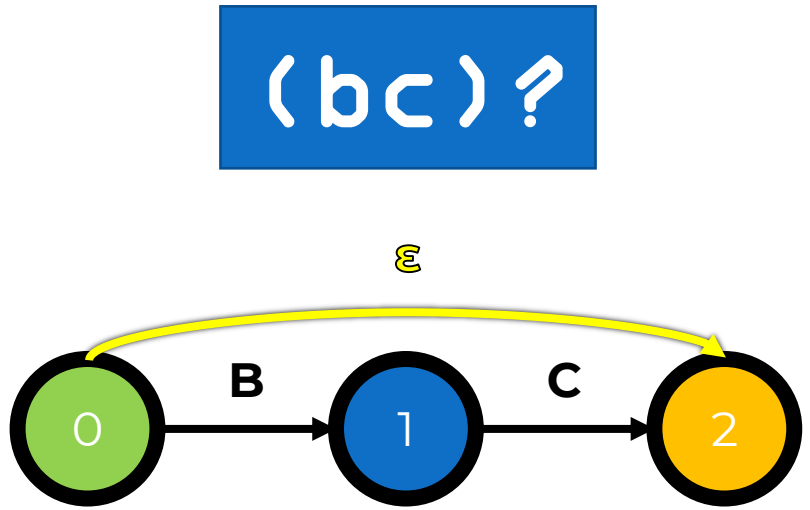
(aba)+



- Для **(...)*** добавить два ε-перехода из стартового состояния в конечное и обратно:

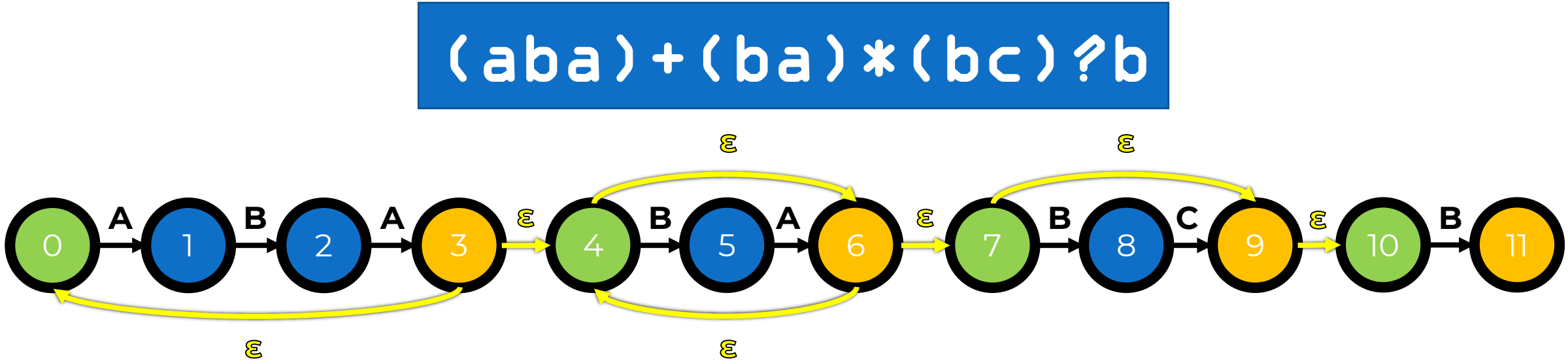


- Для **(...)?** добавить ε-переход из стартового состояния в конечное:



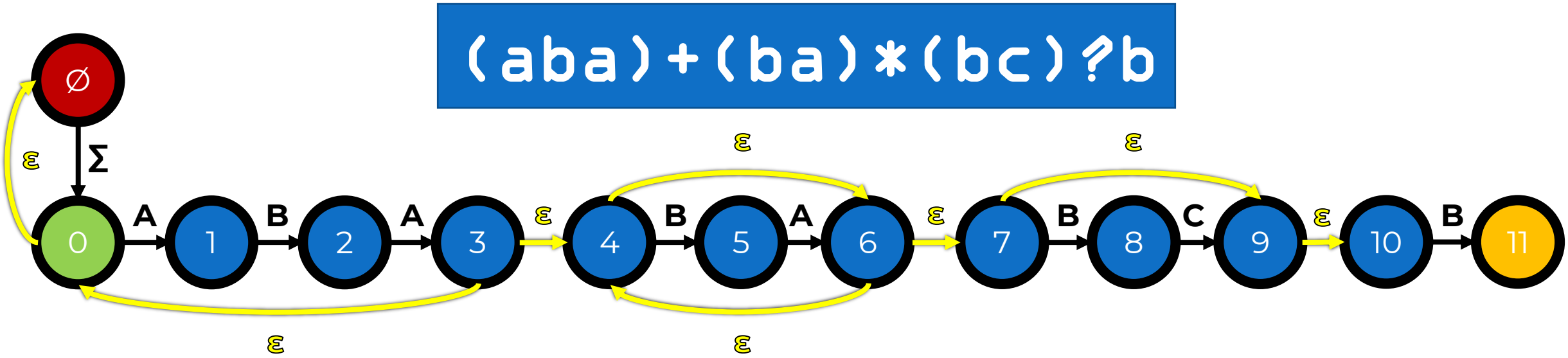
Построение регулярного выражения

3. Соединяем автоматы. Начальное состояние остается начальным только у первой части, конечное остается конечным только у последней.



Построение регулярного выражения

4. Если хотим искать паттерн во всем тексте – добавить специальное состояние Ø с переходом в стартовое состояние по любому символу.



Асимптотика недетерминированного конечного автомата

За какое время можно проверить, что в последовательности есть паттерн?

$O(M) \cdot N = O(MN)$

А если нужно найти все вхождения паттерна?

$O(MN\alpha^N)$

Спасибо за внимание!