Алгоритмы и структуры данных

Суффиксное дерево

CS Center, Новосибирск

Online поиск шаблонов в тексте

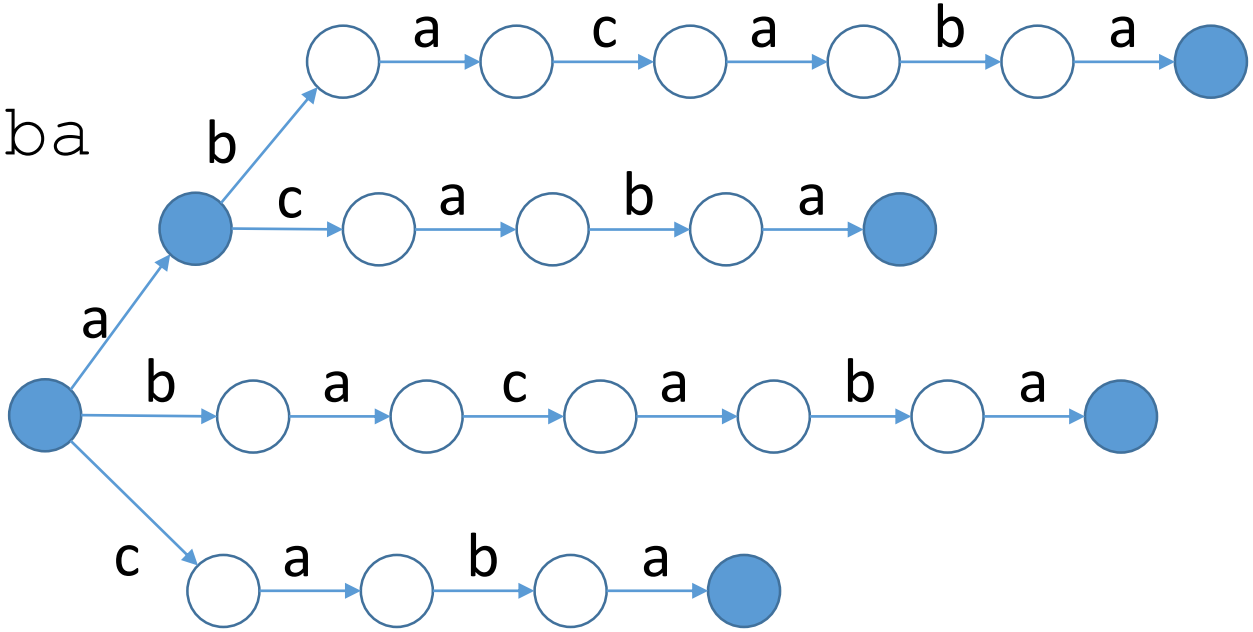
- Дан текст T
  - Можно его предобработать, O(|T|) на предобработку
- Шаблоны P_i приходят один за другим
  - Хочется отвечать на запрос за O($|P_i|$)

- Как предобработь текст?
  - Подстрока – это префикс суффикса
  - Нужно выяснить: P – префикс какого-либо суффикса T?
    - Как эффективно закодировать все суффиксы T?

Идея: бор на суффиксах T

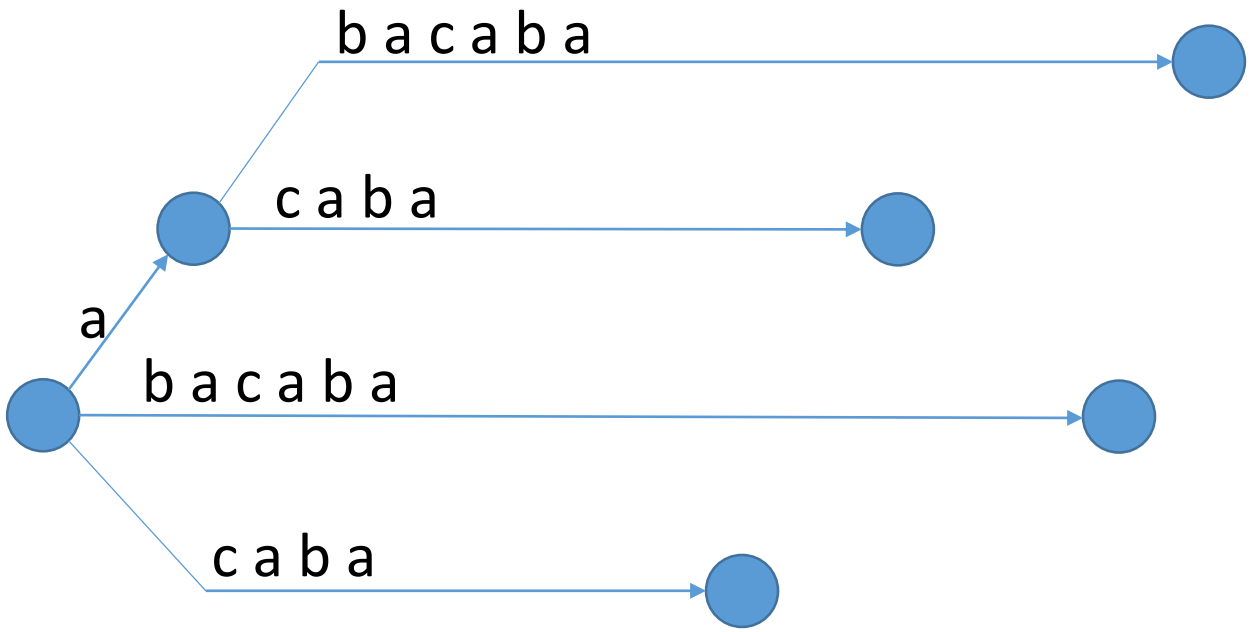
Пример

- T = `abacaba`



- Размер $O(|T|^2)$
  - Как уменьшить?
    - Много лишних вершин

Идея: сжатый суффиксный бор

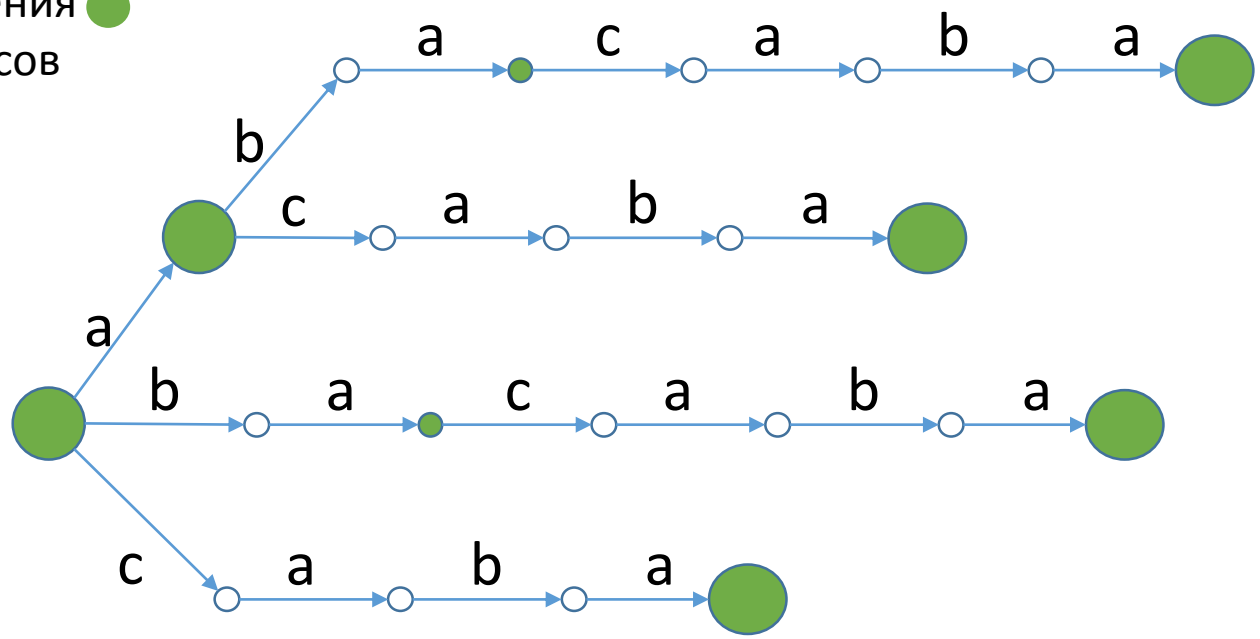


- Суффиксное дерево
  - Не более N листьев => O(N) вершин
- Каждое ребро помечено подстрокой [i : j)

Суффиксное дерево

- Положение в суффиксном дереве:
  - Явное – вершина дерева
  - Неявное – позиция внутри строки на ребре

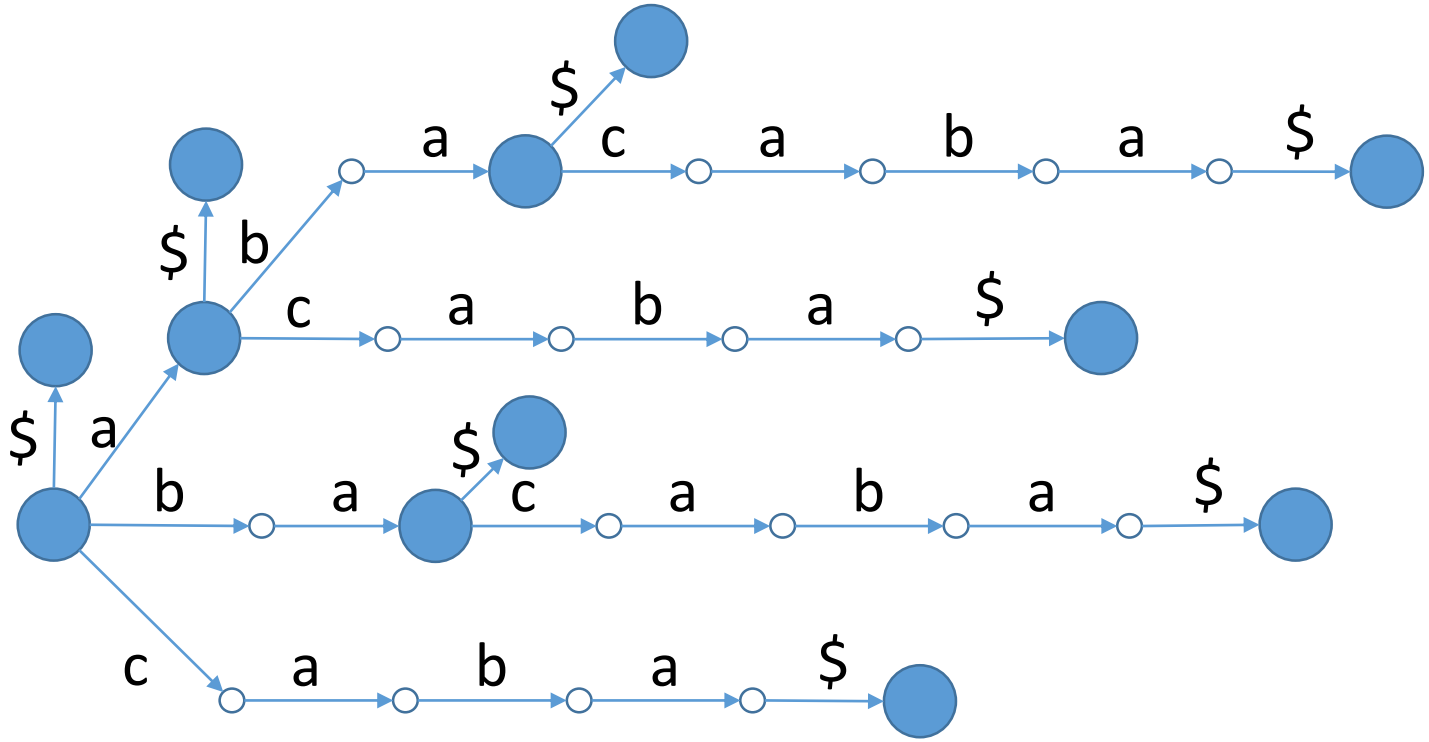
Положения суффиксов



Суффиксное дерево

Сделаем положения суффиксов явными:

- T’ = T + $



Алгоритм Укконена

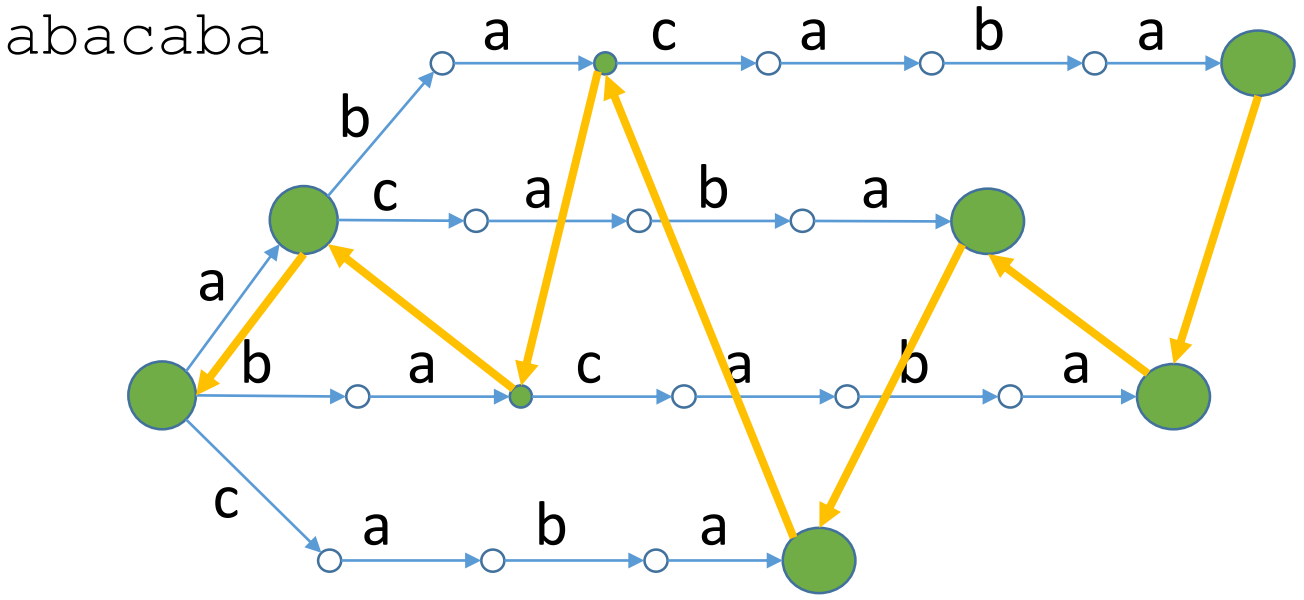
- Начинаем с пустого дерева (для пустой строки)
- На каждом шаге добавляем очередной символ строки
  - На i-ом шаге удлинняются i суффиксов

abac	**abac**	**abac**	**abac**	**abac**
ε	ε+a	a+b	ab+a	aba+c
	ε	ε+b	b+a	ba+c
		ε	ε+a	a+c
			ε	ε+c
				ε

- N таких шагов

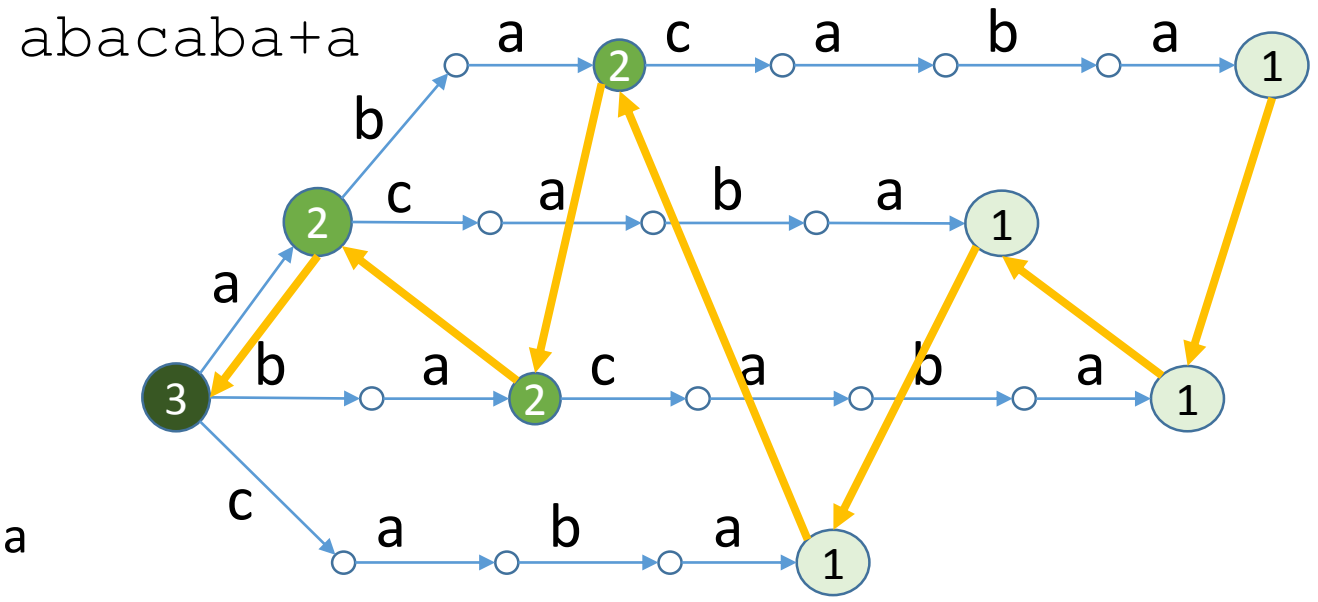
Алгоритм Укконена: добавление символа

- Шаг i: добавляем символ a к i суффиксам
  - Требуется добавить его в i положений в дереве
    - Перебираем эти положения в порядке уменьшения глубины

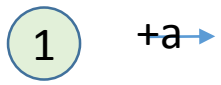


Алгоритм Укконена: добавление символа

- Шаг i: добавляем символ a к i суффиксам
  - Требуется добавить его в i положений в дереве
    - Перебираем эти положения в порядке уменьшения глубины



- Типы положений:



Лист

Положение без символа a (явное или нет)

Положение с символом a (явное или нет)

Алгоритм Укконена: типы положений

- Типы:

1 +a→ Лист

2 +a→ / нет a→ Положение без символа a

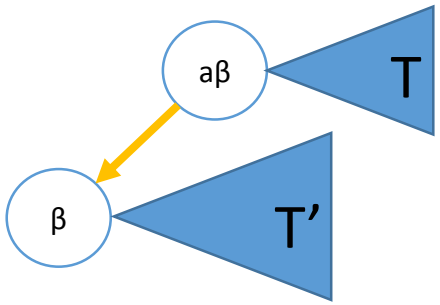
3 a→ Положение с символом a

Пометка ребра изменяется с [x : i) на [x : i+1)
- Можно удлиннять строку на ребре неявно
  - Не требуется изменять дерево

Добавляет новую вершину
- O(N) суммарно по всем итерациям

Не требуется изменять дерево

- Перебираем положения в порядке уменьшения глубины
  - Номера типов положений не убывают: «1, 1, ... 1, 2, 2, ... 2, 3, 3, ... 3»

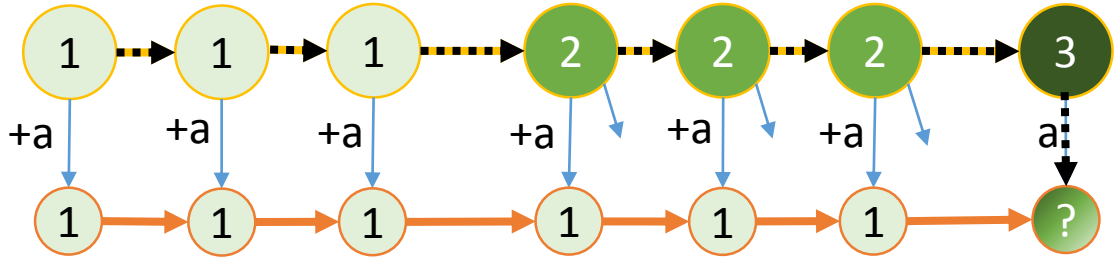


γ ∈ T => γ ∈ T’

Пришли в положение типа 3 =>
все следующие положения не требуют изменений
- можно заканчивать итерацию

Алгоритм Укконена: изменение типов положений

Положения на итерации i



Положения на итерации i+1

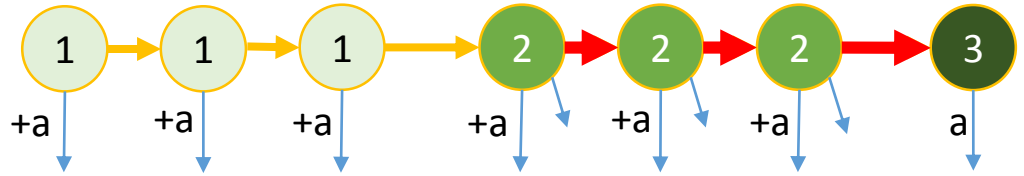
Потенциал := символьная глубина

- уменьшается на 1 при переходе к следующему положению
- растёт на 1 при переходе на следующую итерацию
  - Растёт не более N раз => уменьшается не более N раз
    - O(N) положений посетим в сумме по всем итерациям

Алгоритм Укконена: переход от положения к положению

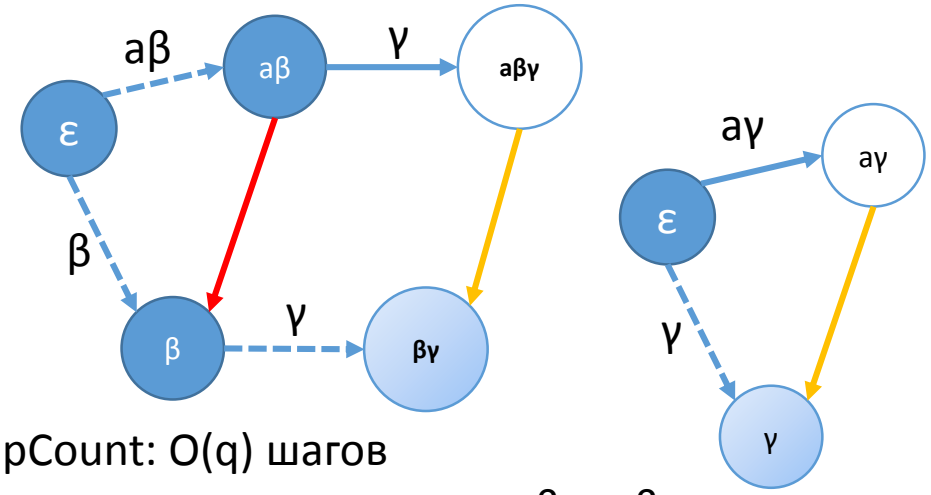
- Храним суффиксные ссылки для внутренних вершин дерева

  « aβ ---> β »



  - При создании вершины запомним её суффиксную ссылку

- Как вычислять суффиксные ссылки для произвольных положений?



SkipCount: O(q) шагов

q – число вершин на пути от β до βγ

Потенциал := вершинная глубина

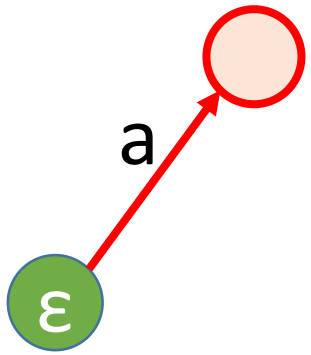
- при SkipCount потенциал растёт не менее, чем на q-1
- при переходе на следующую итерацию потенциал не уменьшается
- максимум потенциала = N
- N шагов, на каждом потенциал уменьшается не более, чем на 1
- => сумма q по всем шагам = O(N)

Алгоритм Укконена: пример

T = abacaba$

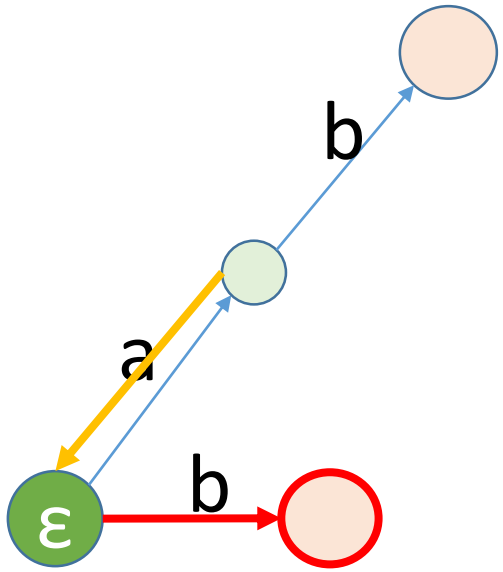
Алгоритм Укконена: пример

T = abacaba$



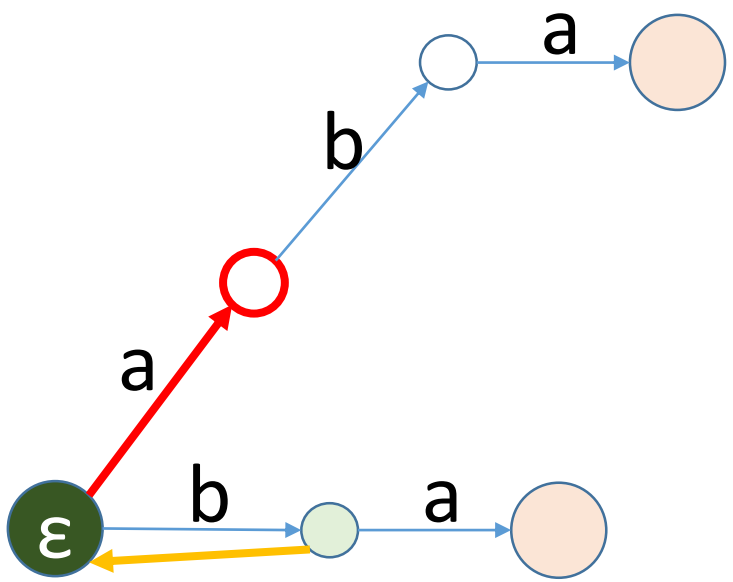
Алгоритм Укконена: пример

T = abacaba$



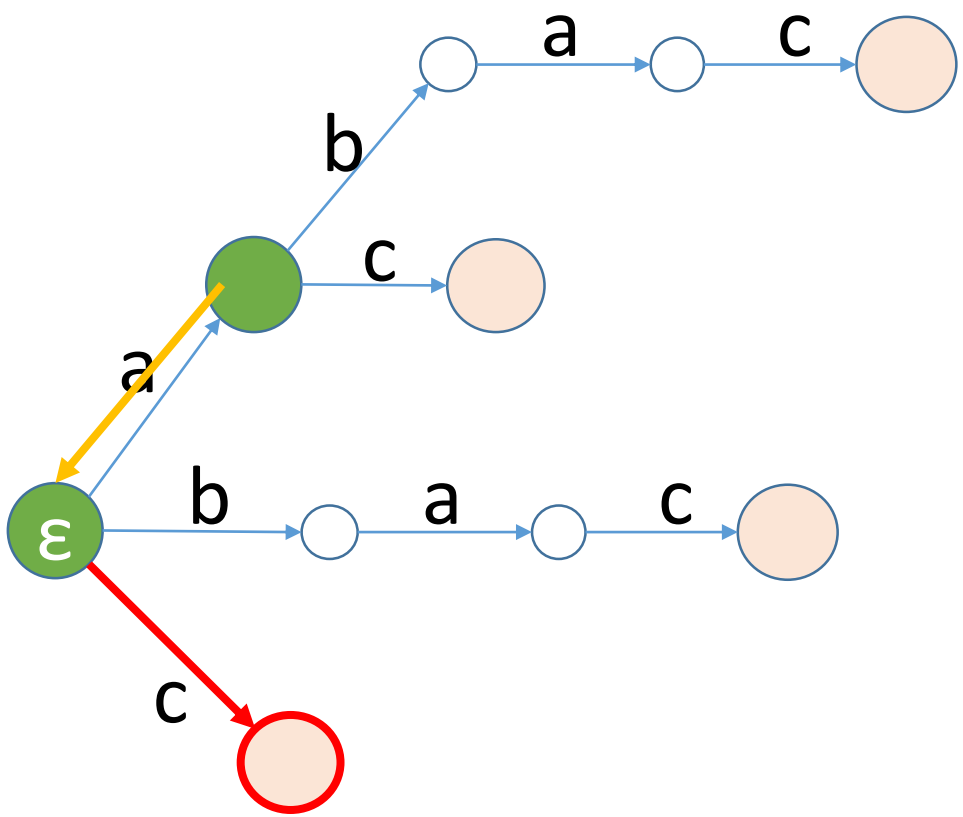
Алгоритм Укконена: пример

T = abacaba$



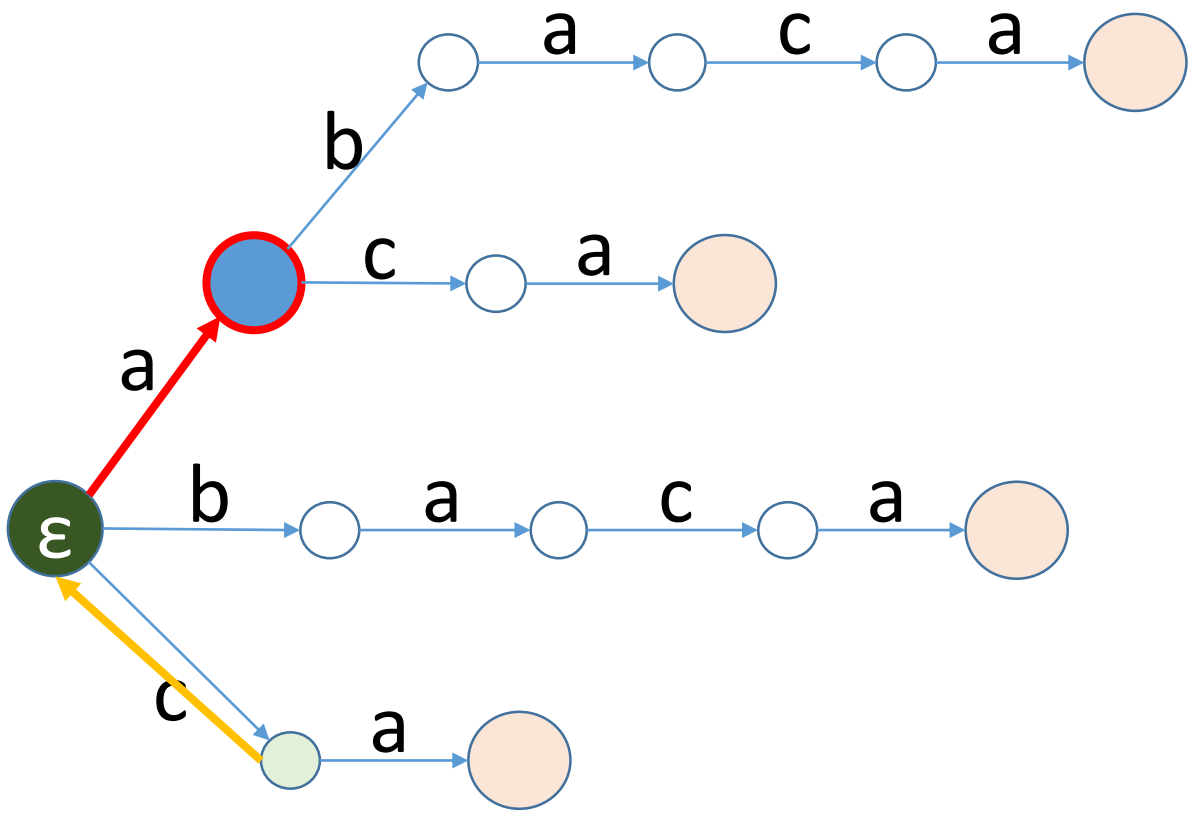
Алгоритм Укконена: пример

T = abacaba$



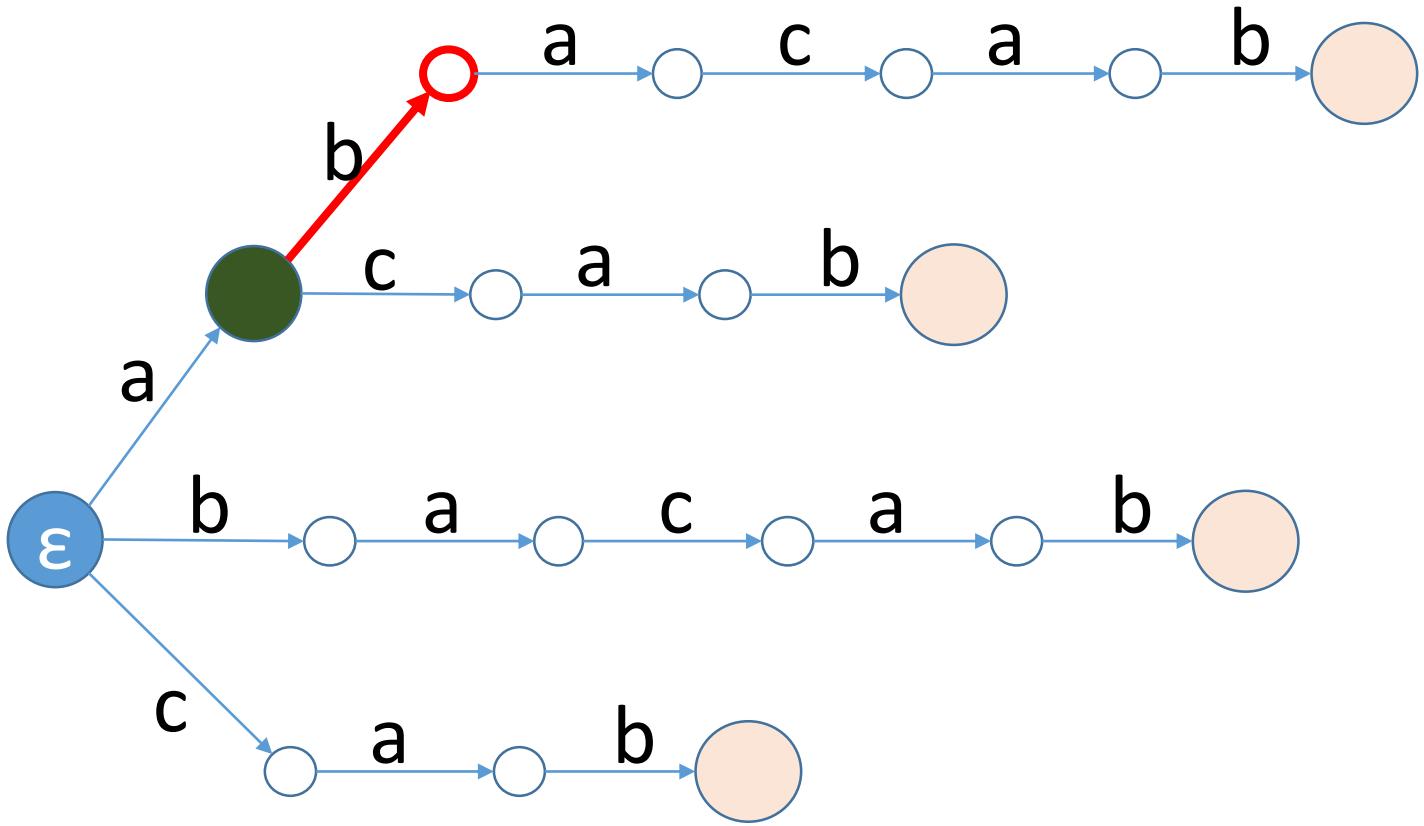
Алгоритм Укконена: пример

T = abacaba$



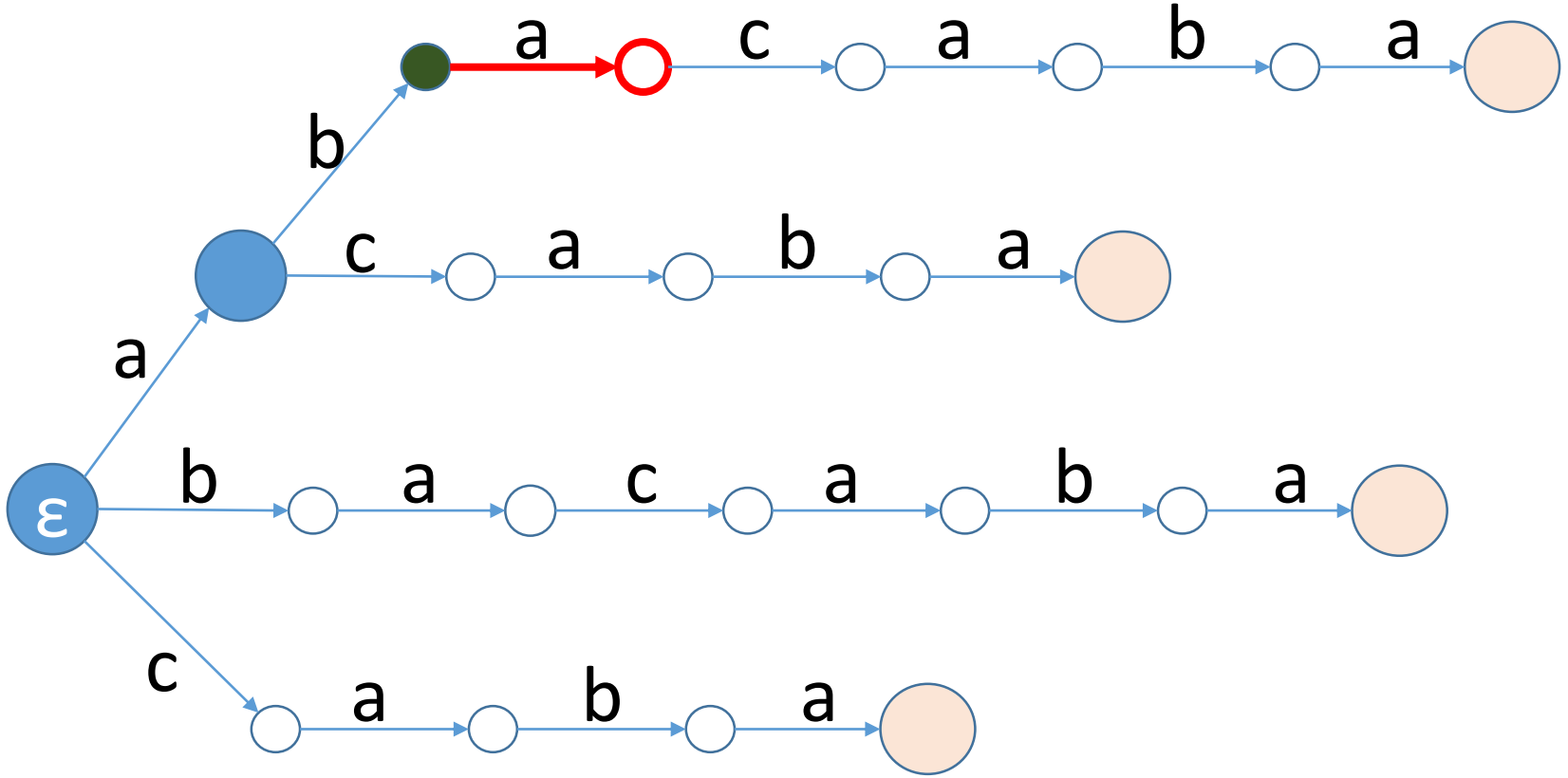
Алгоритм Укконена: пример

T = abacaba$



Алгоритм Укконена: пример

T = abacaba$



Алгоритм Укконена: пример