Алгоритмы и структуры данных

Префикс-функция, поиск бора

CS Center, Новосибирск

Периодичность и borders

Пусть $\alpha = \beta\ \beta\ \ldots\ \beta\ \beta'$ и β' $_{is\_prefix}$ β

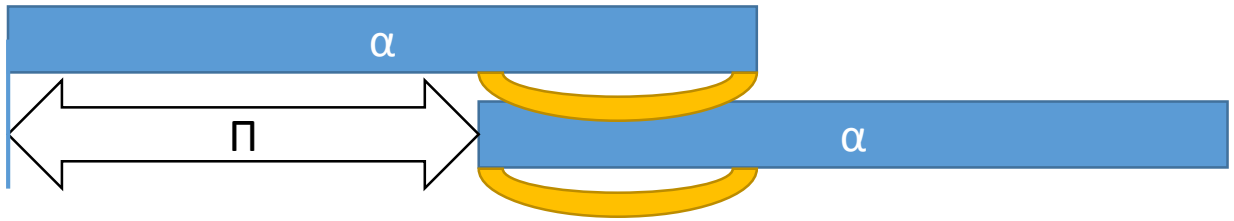
- значит $|\beta|$ - период α

П – период α, если:

- $\alpha[\ П :) = \alpha[: |\alpha| - П\ )$

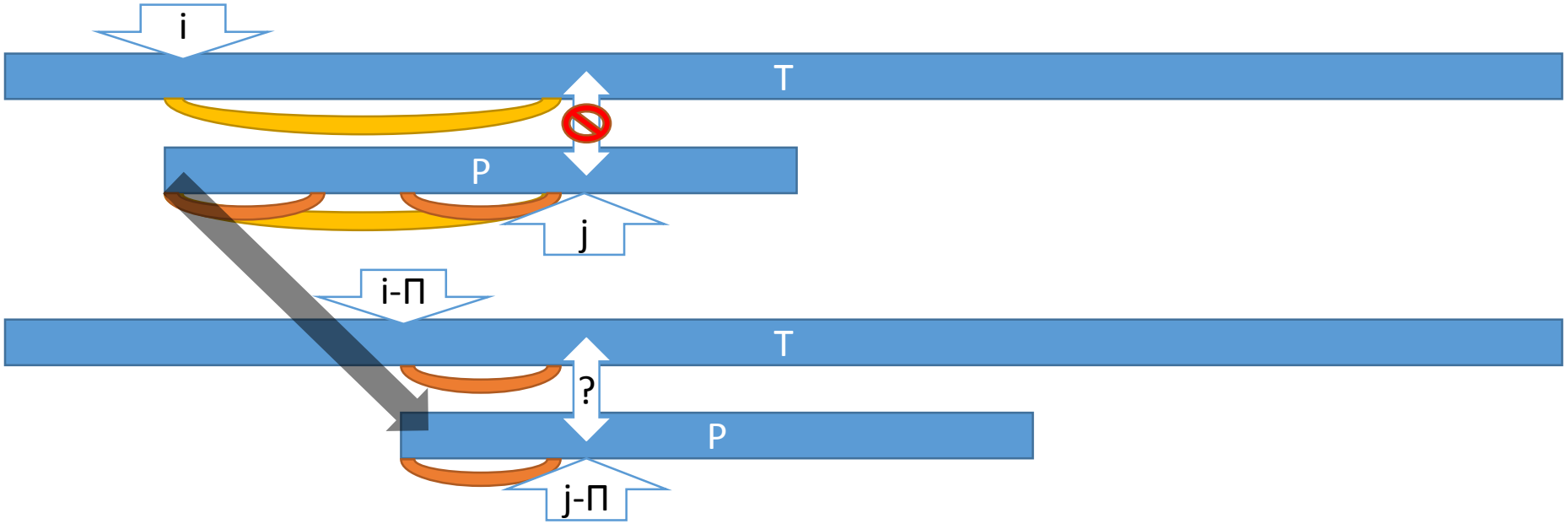
γ $_{is\_border}$ α, если γ $_{is\_prefix}$ α и γ $_{is\_suffix}$ α

- $\gamma = \alpha[\ П :) = \alpha[: |\alpha| - П\ )$ $\qquad\qquad |\gamma| = |\alpha| - П$



Снова поиск шаблона в тексте

P, T ∈ Σ*, P $_{is\_sub}$ T?



П – минимальный собственный период P[: j)

- Можно предподсчитать для все П[j]
  - тогда поиск за O(T)

Префикс-функция строки

- $\Pi_\alpha[\, j \,]$ = длина макс. бордера $\alpha[\, : j \,)$, $1 < j \leq |\alpha|$

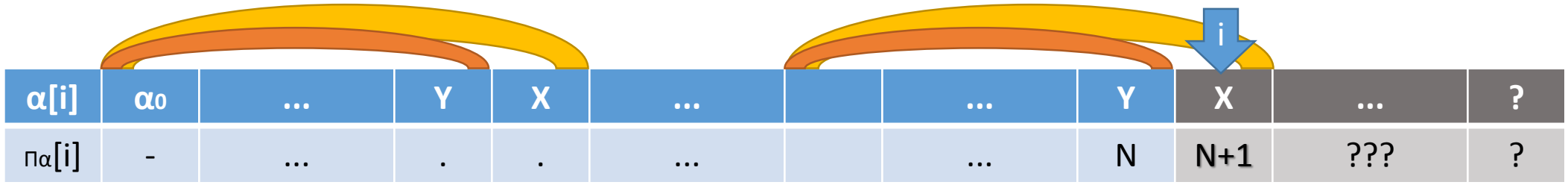
Пример:

α = abacababaca

α[i]	a	b	a	c	a	b	a	b	a	c	a
Πα[i]	0	0	1	0	1	2	3	2	3	4	5

Префикс-функция за линейное время

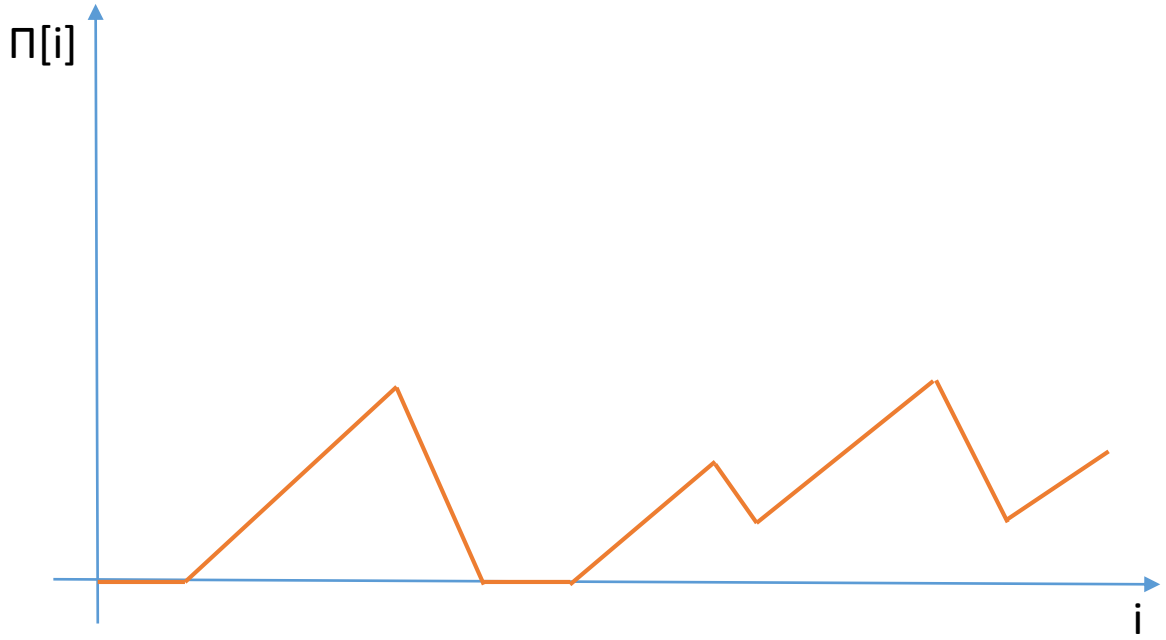
- $\Pi_\alpha[\ j\ ]$ = длина макс. бордера $\alpha[\ :\ j\ )$, $1 < j \le |\alpha|$



- γ is_border β, β is_border α => γ is_border α
- $\{\ \Pi[\ j-1\ ],\ \Pi[\ \Pi[\ j-1\ ]-1\ ],\ \Pi[\ \Pi[\ \Pi[\ j-1\ ]-1\ ]-1\ ],\ \ldots,\ 0\ \}$ – длины всех бордеров $\alpha[\ :\ j\ )$
  - перебираем эти значения, пока не выполнится $\alpha[p] = \alpha[i]$

Префикс-функция за линейное время

```
П[ 0 ] = 0
for i ∈ [ 1 : |α| )
    p = П[ i-1 ]
    while p > 0 && α[ p ] != α[ i ]
        p = П[ p-1 ]
    П[ i ] = (α[ p ] == α[ i ]) ? p+1 : 0
```



- O(N) итераций внешнего цикла
- в сумме O(N) итераций внутреннего цикла
  - П растёт не более N раз и уменьшается не более N раз

Алгоритм Кнута-Морриса-Пратта

- α = P $ T, $ - sentinel
  - $\Pi_\alpha[\ j\ ]$ = |P| => вхождение с позиции j-2*|P|

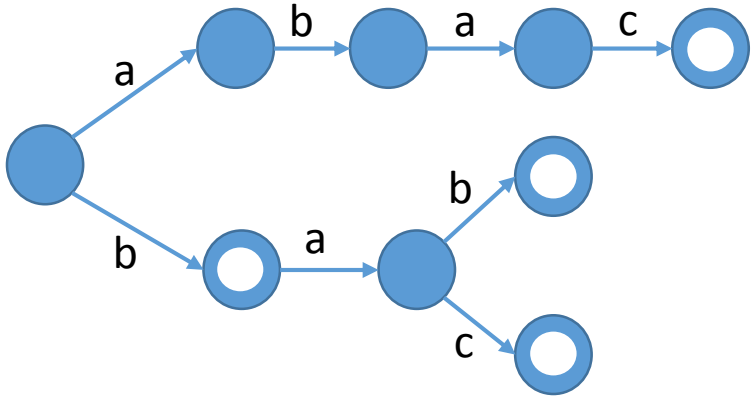
- Все значения Π_α не превосходят |P|
  - Можно ограничиться O(|P|) дополнительной памяти
  - Вычисление первых |P| значений Π_α – предобработка шаблона

Поиск множества образцов

- Желаемая сложность: Σ|Pi| + |T| + A (A – число ответов)

- Бор (trie) – корневое дерево для хранения набора строк
  - рёбра помечены символами
  - помечены конечные вершины

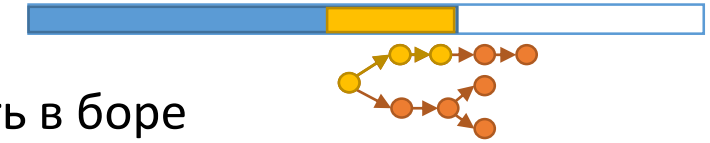
- Пример:
  - { abac, b, bab, bac } ->



Алгоритм: идея

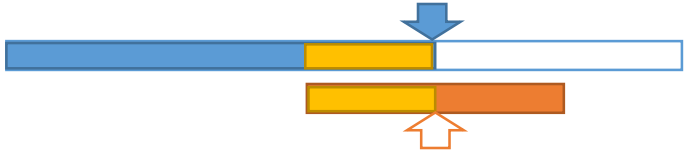
- Построим бор для заданных шаблонов
- Проходим по тексту и перемещаемся по бору
  - Вспомним: КМП вычисляет длиннейший префикс шаблона равный суффиксу прочитанной части текста

  - Аналогия для множества шаблонов:
    - длиннейший префикс шаблона -> длиннейший путь в боре



    - нужно на каждом шаге вычислять такой путь в боре

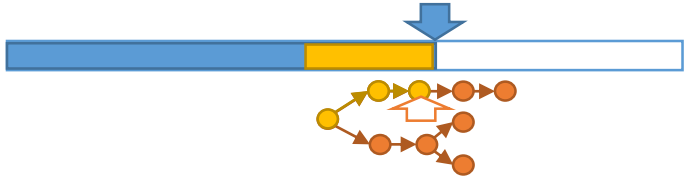
Алгоритм: идея



Один шаблон – КМП

Состояние – позиция в шаблоне

- буква совпадает => продвишаемся по шаблону
- иначе => откатываемся по шаблону, используя префикс-функцию



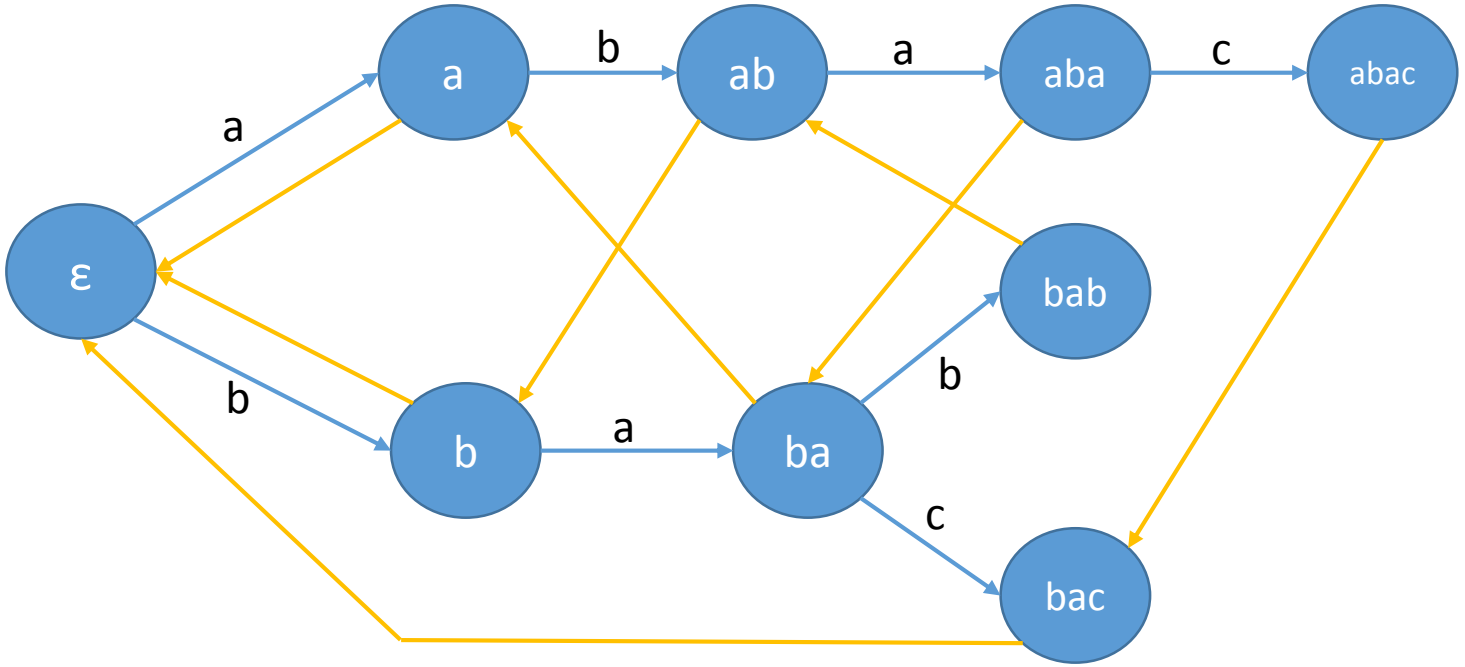
Множество шаблонов

Состояние – вершина бора

- есть ребро, помеченное буквой => продвигаемся по бору
- иначе => откатываемся по бору, используя префикс-функцию на боре

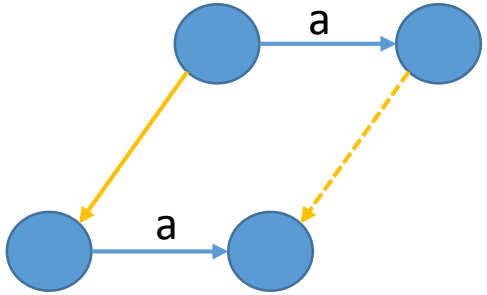
Префикс-функция на боре

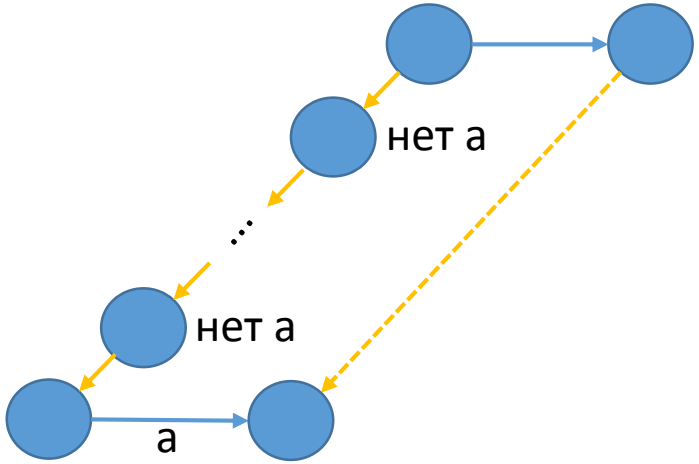
- Соотвествтвие: вершины бора <-> строки
- П(α) = β: β есть в боре, β $_{is\_suff}$ α, |β| < |α|, |β| максимальна

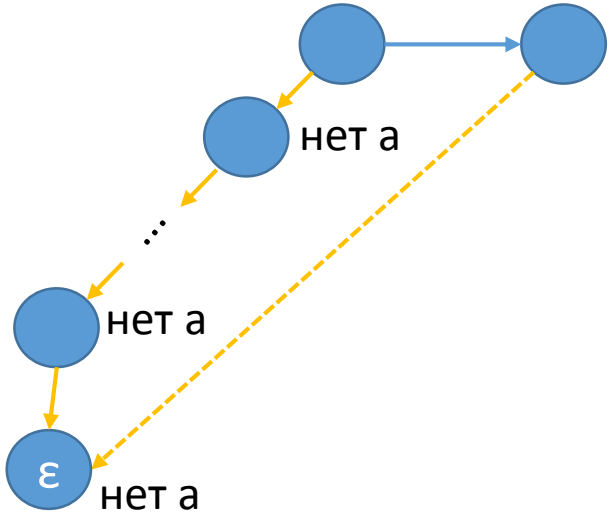


Префикс-функция на боре: вычисление

- { α, П(α), П(П(α)), ..., ε } = { β есть в боре, β $_{is\_suff}$ α }
- Порядок вычисления – по увеличению длины строки
  - порядок обхода дерева в ширину







Префикс-функция на боре: сложность вычисления

- e = (u,v)
  - cost(e) – сколько времени вычисляли П(v)
- Σ_e cost(e) = T – сложность вычисления
- $\Sigma_{v\text{-leaf}}$ cost(path(e,v)) = S ≥ T
- Докажем, что $S \le \Sigma|P_i|$:
  - cost(path(e,v)) ≤ |v| - аналогично префикс-функции для строки
- Значит $T \le \Sigma|P_i|$ – линейная сложность вычисления

Алгоритм Ахо-Корасик

- P:

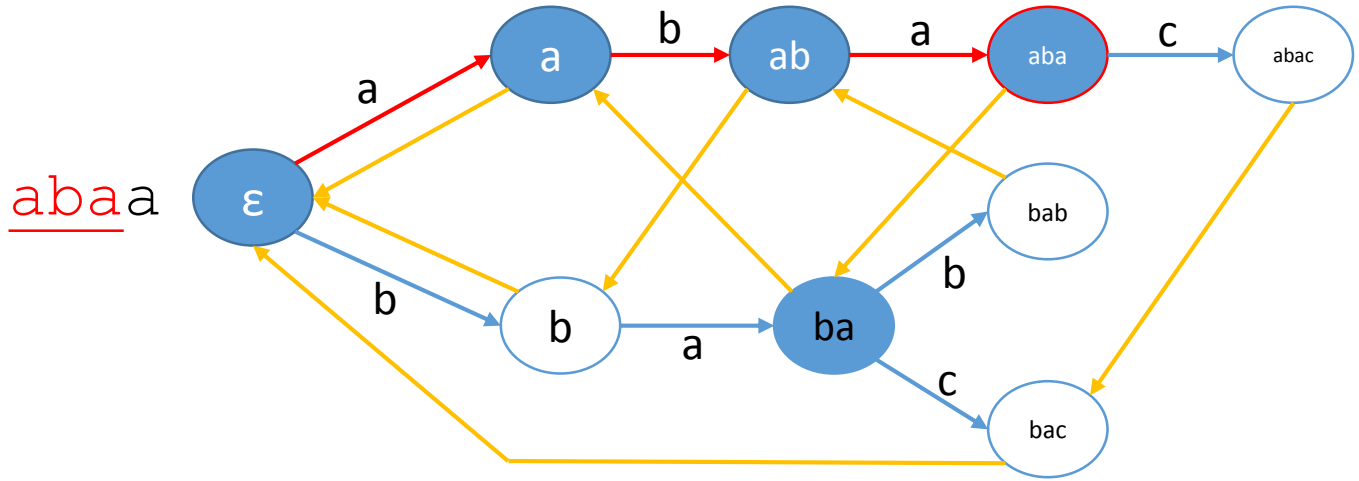
  abac

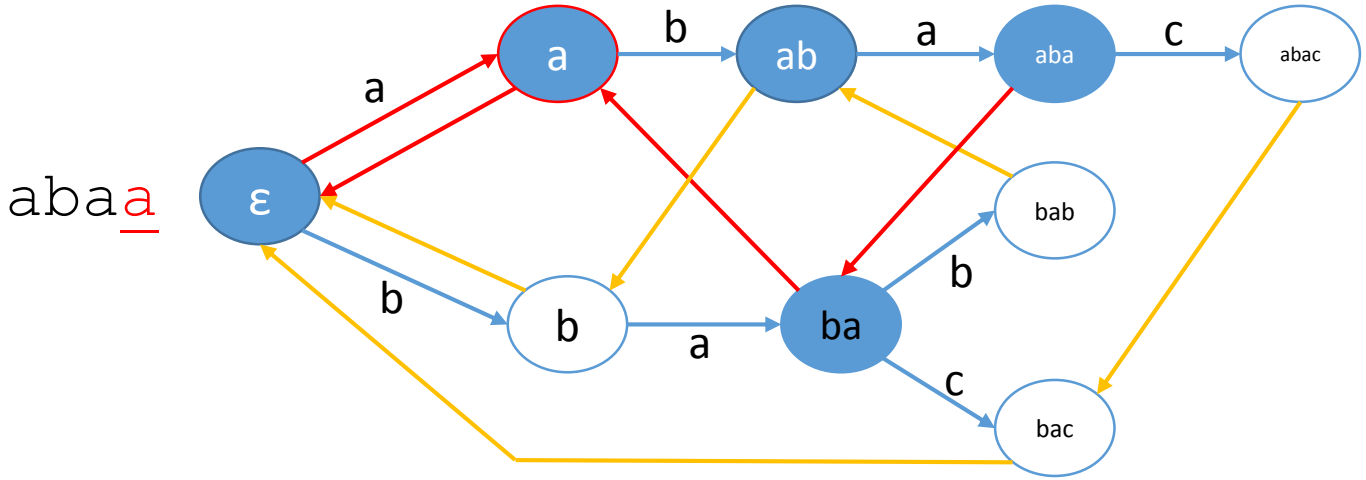
  b

  bab

  bac

- T = abaa





Алгоритм Ахо-Корасик

- Подсчитать общее количество вхождений
  - В каждой вершине бора предподсчитаем число шаблонов, являющихся суффиксами строки
- Найти вхождения
  - В каждой вершине сделаем ссылку на длиннейший шаблон, являющийся собственным суффиксом строки

Бор: накладные расходы

M – сумма длин шаблонов

- Массив ссылок в вершине:
  - Память O(|Σ|*M), время перехода O(1)
- Дерево поиска:
  - Память O(M), время перехода O(log|Σ|)