# Второй курс, осенний семестр 2018/19
# Практика по алгоритмам #6

## Суффиксный автомат

## 10 октября

Собрано 11 октября 2018 г. в 18:12

---

## Содержание

# Суффиксный автомат

1. **Простой суфавтомат**
   Суффиксный автомат строки $s$ – минимальный ДКА, принимающий все суффиксы $s$, и только их. Постройте его за $|s|^2$ или за $\frac{|s|^3}{\omega}$.
2. **Строки в вершине**
   $M(v)$ – максимальная по длине строка, по которой состояние $v$ достижимо из стартового, а $m(v)$ – минимальная такая строка. Докажите, что
   a) $m(v)$ – суффикс $M(v)$
   b) все суффиксы $M(v)$, не короче $m(v)$, ведут в $v$
3. **Суффиксные ссылки**
   $suf(v)$ – состояние, в которое ведёт наибольший суффикс $M(v)$, который не ведёт в $v$, или -1 для $v = 0$. Докажите, что суффиксные ссылки автомата строки $s$ образуют структуру суффиксного дерева строки $\bar{s}$, и постройте само дерево.
4. **Линейный размер**
   Докажите, что при константом размере алфавита в суфавтомате $\mathcal{O}(|s|)$ состояний и переходов.
5. **Построение**
   Рассмотрим алгоритм построения суфавтомата. Пусть построен автомат для $s$, построим его для $s + c$, где $c$ – очередной символ строки.
   $last$ – состояние, в которое ведёт $s$. Добавим новое состояние $new$.

```
p = last
while (p >= 0 && to[p][c] == -1):
  to[p][c] = new
  p = suf[p]
if (p != -1):
  q = to[p][c]
  if (|M(p)| + 1 == |M(q)|):
    suf[new] = q
  else:
    qc = clone(q) // full copy of state q with edges and suf
    M(qc) = M(p) + c
    suf[q] = qc
    suf[new] = qc
    while (p >= 0 && to[p][c] == q):
      to[p][c] = qc
      p = suf[p]
last = new
```

   Терминальные состояния – достижимые по суфссылкам из $new$. Докажите, что построенный автомат
   a) принимает все суффиксы $s + c$ и только их
   b) минимален
6. **Линейное время**
   Докажите, что добавление в автомат всех символов строки $s$ занимает $\mathcal{O}(|s|)$ времени.

7. **Подсчёт различных подстрок**
   a) Найти число различных подстрок строки.
   b) Найти число различных подстрок у каждого префикса строки.
   c) Найти подстроку данной строки, встречающуюся максимальное число раз, среди таких самую длинную.
8. **$k$-й суффикс**
   Найти $k$-й в лексикографическом порядке суффикс строки.
9. **Подсчёт вхождений** Дан текст $t$ и строка $s$. Найдите
   a) Количество вхождений $s$ в $t$.
   b) Позицию первого вхождения $s$ в $t$.
   c) Позиции всех вхождений $s$ в $t$ за $\mathcal{O}(|s| + |ans|)$.
10. **(*) Неконстантный алфавит**
    Докажите, что при любом алфавите в суфавтомате $\mathcal{O}(|s|)$ состояний и переходов.

# Разбор задач практики

1. **Простой суфавтомат**
   Для каждого суффикса сделаем следующее: будем хранить маску позиций, в которых может заканчиваться каждый его префикс. Начнём со всех позиций, а дальше оставляем те позиции, в которые можно перейти из предыдущих по очередному символу.
   Пример для строки $abcbc$ и суффикса $bcbc$: $aBcBc \Rightarrow abCbC \Rightarrow abcBc \Rightarrow abcbC$.
   Каждой маске, в которой побывали, сопоставим вершину, переходы между масками по очередному символу суффикса. Терминальные – те, которые содержат последний символ. Любой суффикс допускается (очевидно). Любая допускаемая строка – суффикс, потому что есть позиция, с которой мы начали и закончили в последней. Автомат минимален, потому что все правые контексты различны (контекст задаётся маской).
   А ещё можно построить бор и минимизировать его.

2. **Строки в вершине**
   a) Состояние одно и тоже, значит правые контексты $M(v)$ и $m(v)$ равны, значит любое вхождение $M(v)$ в $s$ заканчивается на $m(v)$, т.к. к ним дописывается один и тот же суффикс.
   b) При удалении символов из начала $M(v)$ правый контекст может только увеличиться, а у $m(v)$ он такой же, значит и у всех промежуточных такой же.

3. **Суффиксные ссылки**
   Развернём суффсылки. На $suf(v)$ напишем развёрнутый префикс $M(v)$ длины $|M(v)| - |m(v)| + 1$.
   Рассмотрим состояние, в которое ведёт какой-то префикс $s$. Откатываясь от него по суффссылкам, мы выписали его целиком в развёрнутом виде, значит любой префикс $s$, т.е. любой суффикс $\bar{s}$, находится в дереве.
   Переход из вершины $v$ в дереве по символу $c$ однозначно определён – это вершина, в которую ведёт строка $c + M(v)$, если такая есть, значит построенное дерево – корректный бор.
   Правый контекст $M(suf(v))$ не равен правому контексту $M(v)$, значит у $M(suf(v))$ есть новые относительно $M(v)$ вхождения. Рассмотрим символ $c$ – первый перед началом нового вхождения $M(suf(v))$). $c$ не равен первому символу $m(v)$ (т.к. вхождение новое), значит у $c + M(suf(v))$ правый контекст отличается и от $m(v)$, и от $M(suf(v))$, значит в $suf(v)$ ведёт ещё и $suf$ из состояния для $c + M(suf(v))$, т.е. в любое состояние входит хотя бы две суффсылки, значит дерево – сжатый бор.

4. **Линейный размер**
   Состояний столько же, сколько вершин в сжатом дереве развёрнутой строки, т.е. линия, а переходов не больше чем число состояний на константу, т.е. тоже линия.

5. **Построение**
   a) Терминальные состояния = суффиксный путь от $new$. По построению в $new$ приводит строка $s + c$ и ещё сколько-то её суффиксов, в $suf(new)$ приводят её суффиксы меньшего размера и т.д.

b) Правые контексты у всех состояний различны. К строкам в правых контекстах старых состояний добавился $c$ в конец, все остались различны. $RC(new)$ единственный равен $\{\varepsilon\}$. $RC(qc) = RC(q) + \{\varepsilon\}$, нигде не встречается, потому что $RC(q)$ уникален. Все состояния, у которых есть $\varepsilon$ находятся на суффиксном пути, значит их правые контексты образуют расширяющуюся последовательность.

6. **Линейное время**
Докажите, добавление в автомат всех символов строки $s$ занимает $\mathcal{O}(|s|)$ времени.
Из линейности размера автомата следует, что на добавление всех рёбер в $new$ и на копирование $q$ мы суммарно потратим линию времени. Осталось понять, почему суммарно перенаправлений из $q$ в $qc$ будет линия.
$M(suf(suf(new)))$ – это суффикс $s + c$. Обозначим за $i$ позицию начала этого суффикса. Если клонирования не было, $i$ не поменялось. Если клонирование было, то каждое перенаправление сопровождалось переходом по суффиксному пути, который начинался в $last$, то есть влекло за собой уменьшение длины очередного суффикса, что эквивалентно увеличению $i$. Перенаправления заканчиваются, когда $to[p][c] = suf(qc) = suf(suf(new))$, значит позиция начала $M(suf(suf(new)))$ теперь равна новому увеличенному значению $i$. Увеличить $i$ больше чем на $n$ суммарно нельзя, значит перенаправлений тоже линия.

7. **Подсчёт различных подстрок**
   a) Любая подстрока – путь в DAG, которым является автомат. Если DAG, можем динамикой посчитать число путей в нём. $d[v] = 1 + \sum_{vu \in E} d[u]$. Ответ $d[0] - 1$.
   b) После добавления очередного $c$ интересно сколько суффиксов новые. Знаем, что $M(suf(new))$ – максимальный, который был раньше, значит добавим к ответу $|M(new)| - |M(suf(new))|$.
   c) Строка встречается столько раз, сколько есть путей до терминальных вершин из состояния, в которое приводит эта строка. Посчитаем динамикой это количество для каждой вершины. Из всех $v$, для которых оно максимально, выберем ту, у которой $|M(v)|$ максимально.

8. **$k$-й суффикс**
Опять посчитаем количество путей до терминальных состояний, затем идём из старта и перебираем рёбра в порядке возрастания буквы на них. Если для очередного соседа $u$ $d[u] > k$, идём в него, иначе вычитаем $d[u]$ из $k$.

9. **Подсчёт вхождений**
Построим суфавтомат для $t$. $go(s)$ – состояние, в которое приходим по $s$.
   a) Количество путей до терминальных состояний из $go(s)$.
   b) Для каждого состояния $v$ найдём максимально далёкое достижимое терминальное $u$, то есть найдём самый длинный суффикс, началом которого мы можем быть. Первое вхождение $s$ – начало найденного суффикса для $go(s)$.
   c) Все вхождения – начала самых длинных суффиксов из предыдущего пункта для всех состояний, из которых достижимо по суффссылкам $go(s)$.

# Домашнее задание

## 3.1. Обязательная часть

Во всех задачах обязательной части можно пользоваться только суфавтоматом.

1. **(1.5) Суффиксный автоматик**
   Постройте минимальный ДКА, который допускает все суффиксы $s$. Оцените размер, докажите минимальность.

2. **(3) Длины различных подстрок**
   Дана строка $s$. Посчитайте за $\mathcal{O}(|s|)$ суммарную длину различных подстрок
   a) **(1.5)** строки $s$
   b) **(1.5)** каждого префикса $s$

3. **(2) Общая подстрока $k$ строк**
   Предложите алгоритм поиска max общей подстроки $k \leqslant 64$ строк за их суммарную длину.
   **(+1)** Решите за суммарную длину для любого $k$.

4. **(3) Ключевые подстроки**
   Дан набор строк $s_i$. Для каждой $s_i$ найдите min по длине подстроку, которая не встречается в других.

5. **(3) Уникальные суффиксы**
   Два запроса:
   a) `addLetter(c)` – дописать в конец строки символ $c$.
   b) `isUnique(len)` – является ли суффикс длины $len$ уникальной подстрокой.

6. **(3) $k$-я общая подстрока**
   Найти $k$-ю лексикографически общую подстроку $s$ и $t$ за $\mathcal{O}(|s| + |t|)$, алфавит константный.

## 3.2. Дополнительная часть

1. **(3) Неконстантный алфавит**
   Докажите, что при любом алфавите в суфавтомате $\mathcal{O}(|s|)$ состояний и переходов.

2. **(3) Три вхождения**
   Дана строка $s$. Найти число строк $t$ таких, что они имеют хотя бы 3 непересекающихся вхождения в строку $s$.

3. **(3) Число различных подстрок**
   К строке $s$, изначально пустой, поступают запросы добавления символа в конец и удаления символа из начала. После каждого запроса скажите число различных подстрок $s$. Суммарно за линию от числа запросов.

4. **(4) Сумма бордеров.**
   Дана строка $S$. Найти за $\mathcal{O}(n)$ сумму $\sum\limits_{i=1}^{n}\sum\limits_{j=i}^{n} B(S[i..j])$.
   Определение $B(S) = \max x\colon S[0..x) = S[n-x..n)$ (бордер строки).