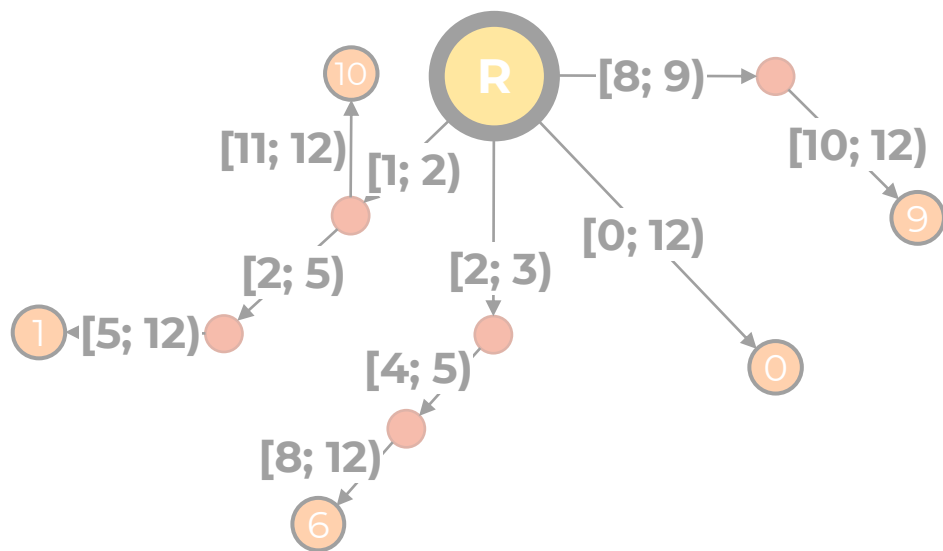




Ищем разные
паттерны в одном
тексте!

Суффиксное дерево

Что у нас
сегодня?

- 1 [Геномные данные](#)
- 2 [Суффиксный бор](#)
- 3 [Суффиксное дерево](#)
- 4 [Суффиксный массив](#)



Какие алгоритмы поиска
паттернов в тексте вы уже
изучили?

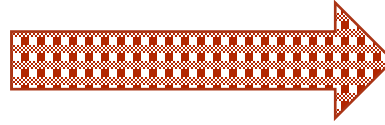
Подумайте, как они
применяются в
биоинформатике.

А может ли стоять
обратная задача?



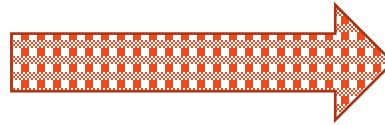
**Да! Если мы, например,
ищем прочтения в геноме.**

Какие есть
подходы
поиска?



Просто искать!

- ❖ Наивный поиск



Предобработать паттерн

- ❖ Автоматы
- ❖ КМП
- ❖ Рабин – Карп



Предобработать текст

- ❖ Суффиксный бор
- ❖ Суфф. дерево
- ❖ Суфф. массив

Суффиксный бор

Для построения суффиксного бора нам необходимо преобразовать текст:

mississippi

Суффиксный бор

1. Добавим в конец слова специальный символ, которого нет в алфавите - \$.

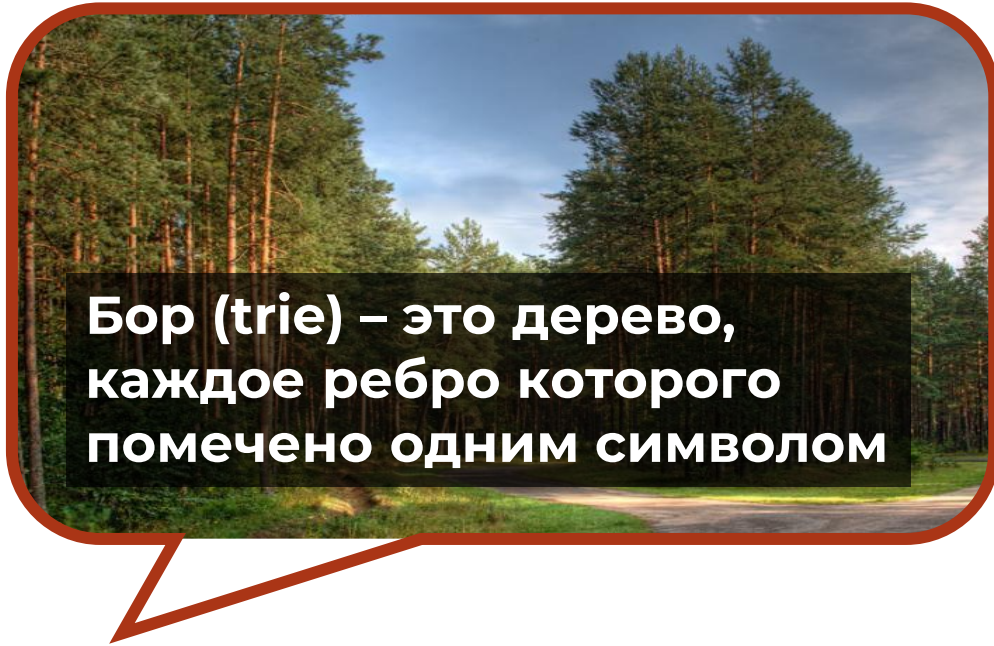
mississippi\$

mississippi\$

2. Выпишем все суффиксы получившегося слова кроме пустого и \$:

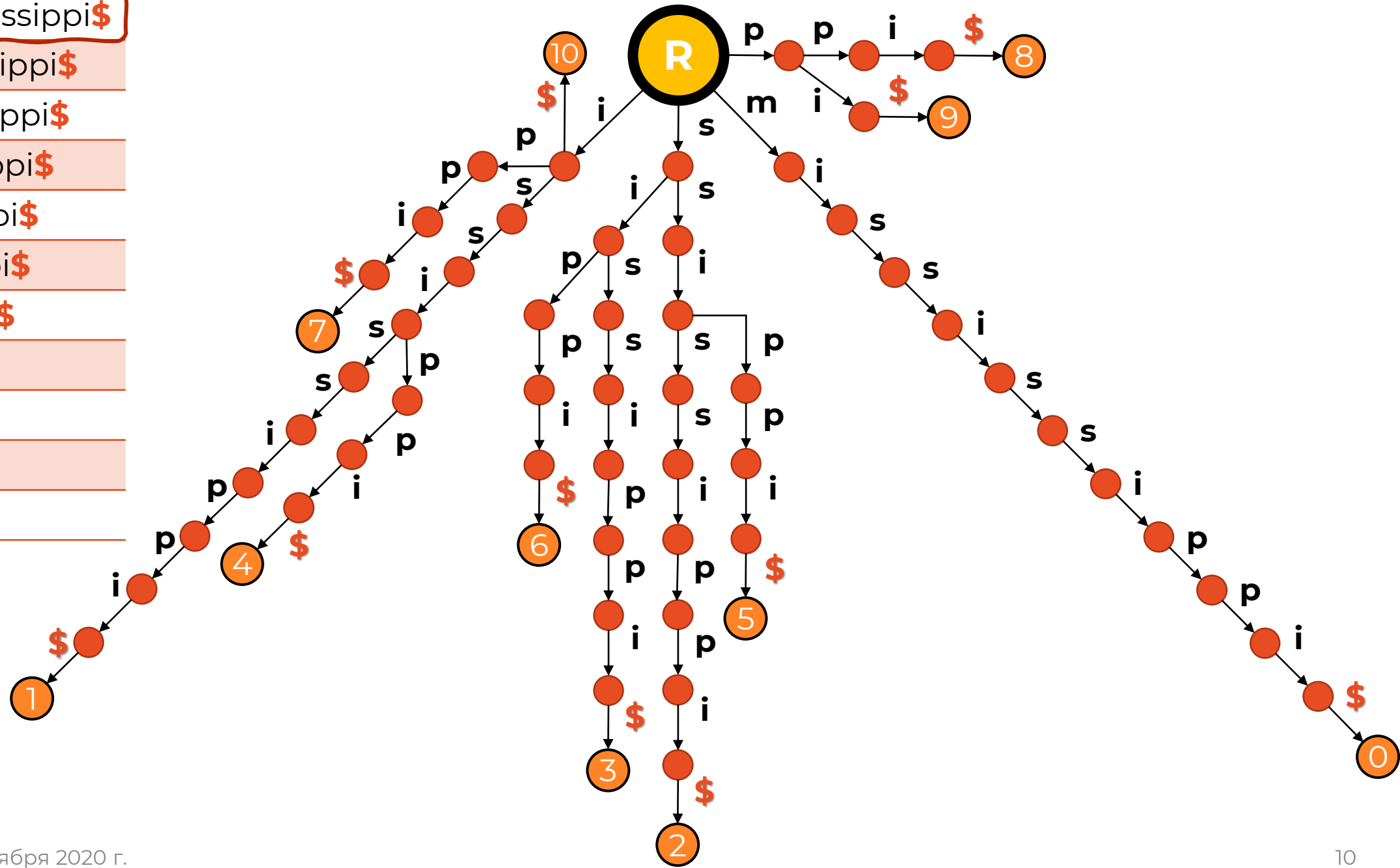
0	mississippi\$
1	ississippi\$
2	ssissippi\$
3	sissippi\$
4	issippi\$
5	ssippi\$
6	sippi\$
7	ippi\$
8	ppi\$
9	pi\$
10	i\$

И теперь по полученной
таблице строим
суффиксный бор!

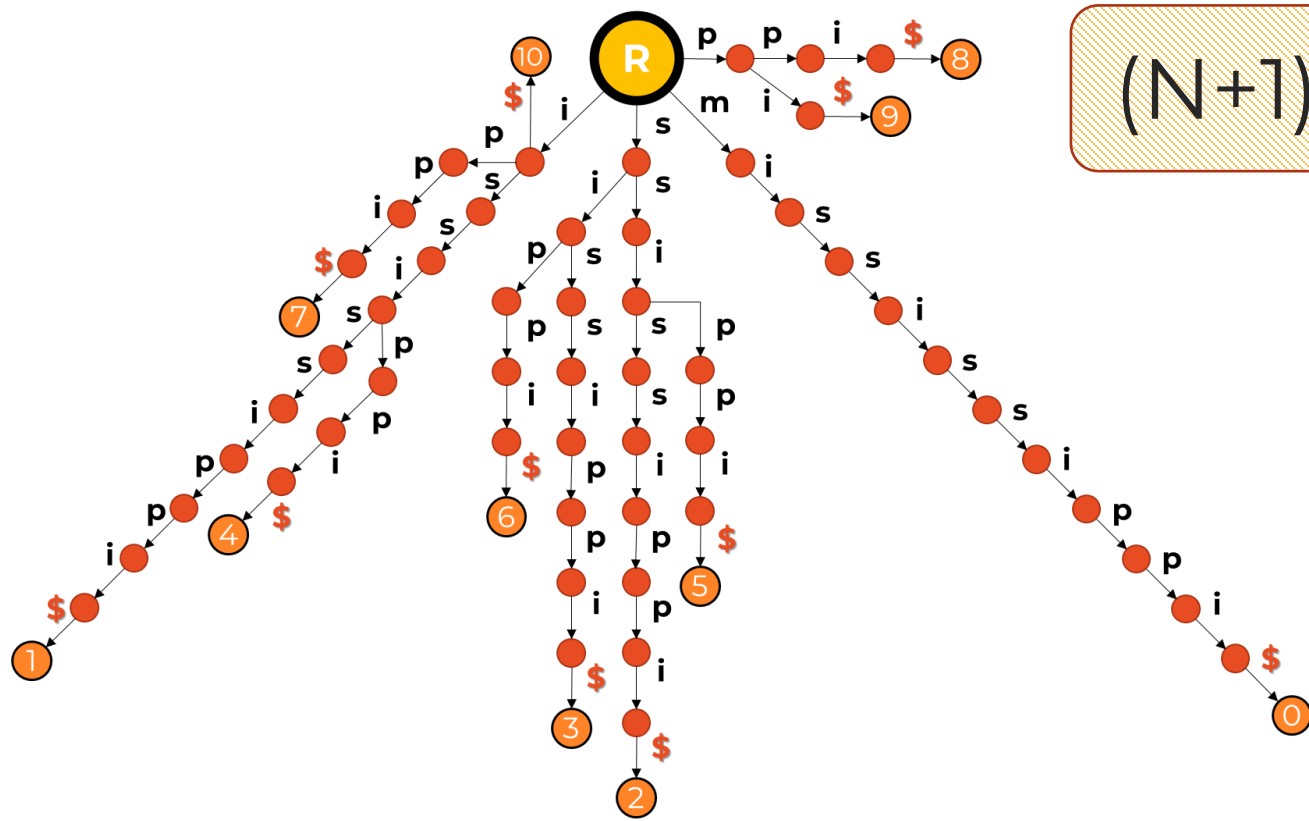


А что такое
бор?

0	mississippi\$
1	ississippi\$
2	ssissippi\$
3	sissippi\$
4	issippi\$
5	ssippi\$
6	sippi\$
7	ippi\$
8	ppi\$
9	pi\$
10	i\$



Сколько вершин
в боре?



$$(N+1) + N + (N-1) + \dots + 2 + 1$$

$$\frac{N(N+1)}{2}$$

Как долго мы строим
суффиксный бор?

Почему?

$O(N^2)$

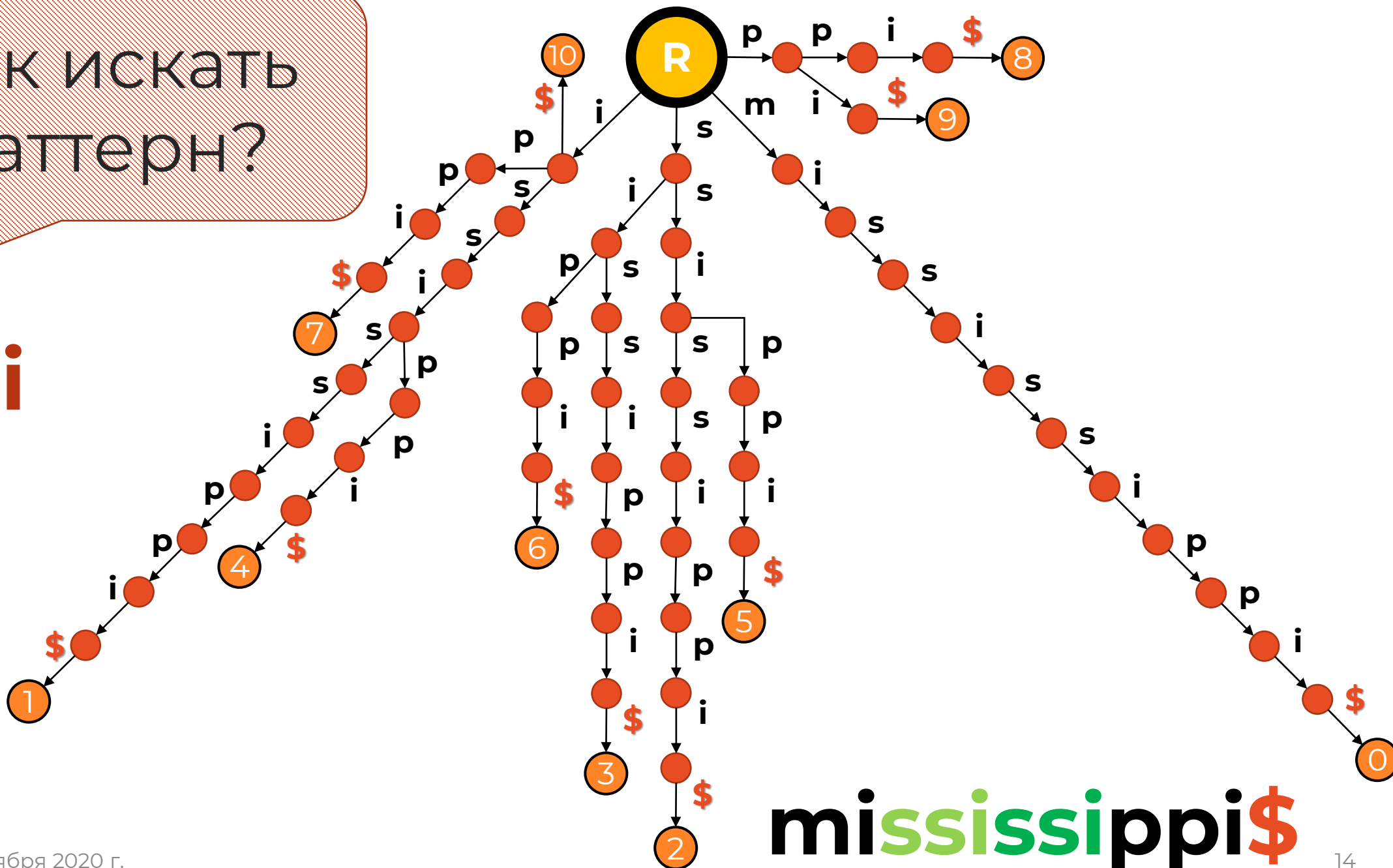
Каждый суффикс мы
добавляем за **$O(N)$** .

А можно ли быстрее?

Нет, размер
нашей структуры
(в среднем) **$O(N^2)$** .

Как искать
паттерн?

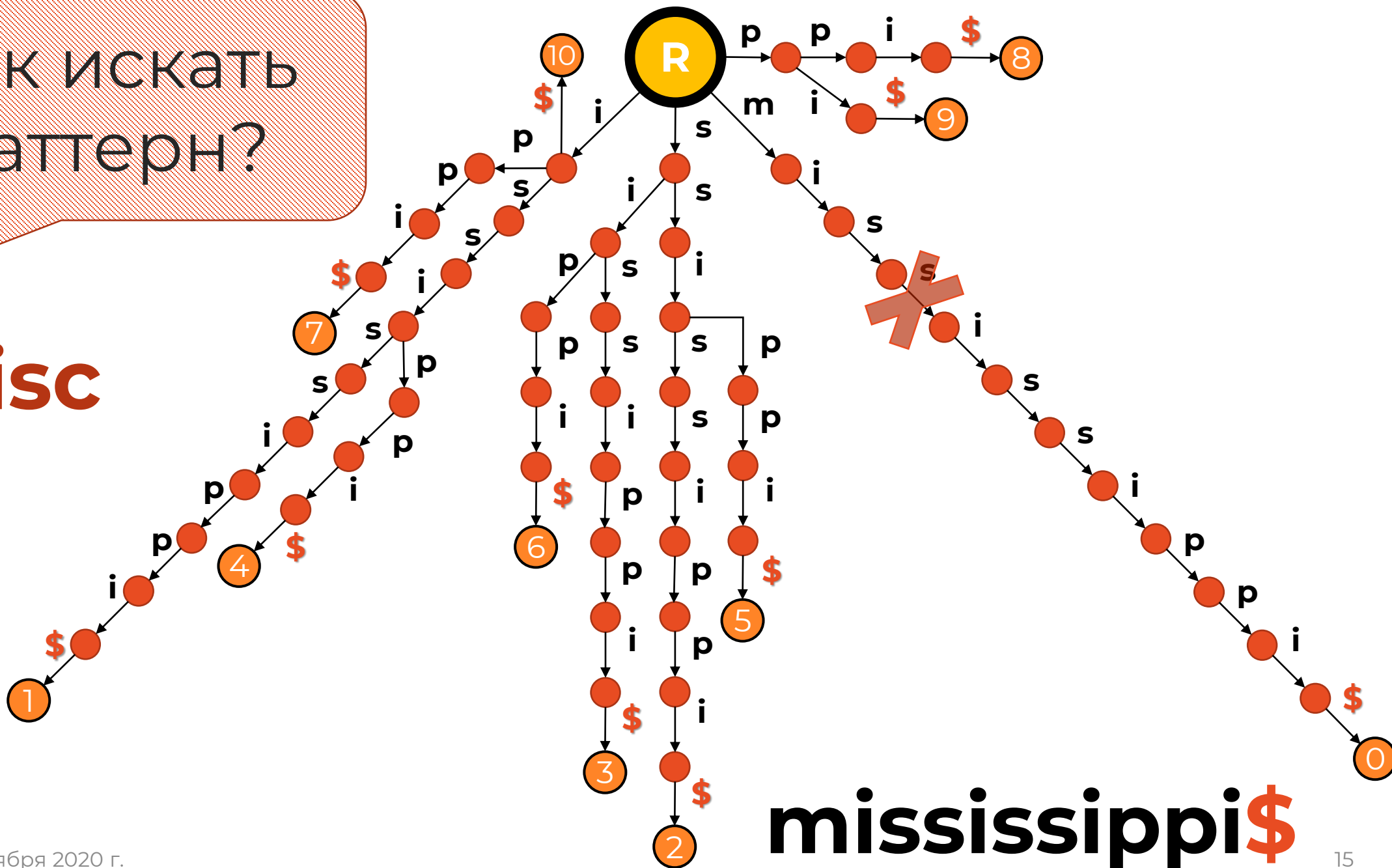
ssi



mississippi\$

Как искать
паттерн?

misc



Сколько ищем паттерн
в плохом случае?

Почему?

$O(M + \alpha(N-M))$.

Нам может понадобиться до α
раз пройти по внутренней
части бора, ее глубина **$O(N-M)$.**

Какова асимптотика
поиска паттерна с
суффиксным бором?

Построение бора

$O(N^2)$

Поиск паттерна

$O(M + \alpha(N-M))$

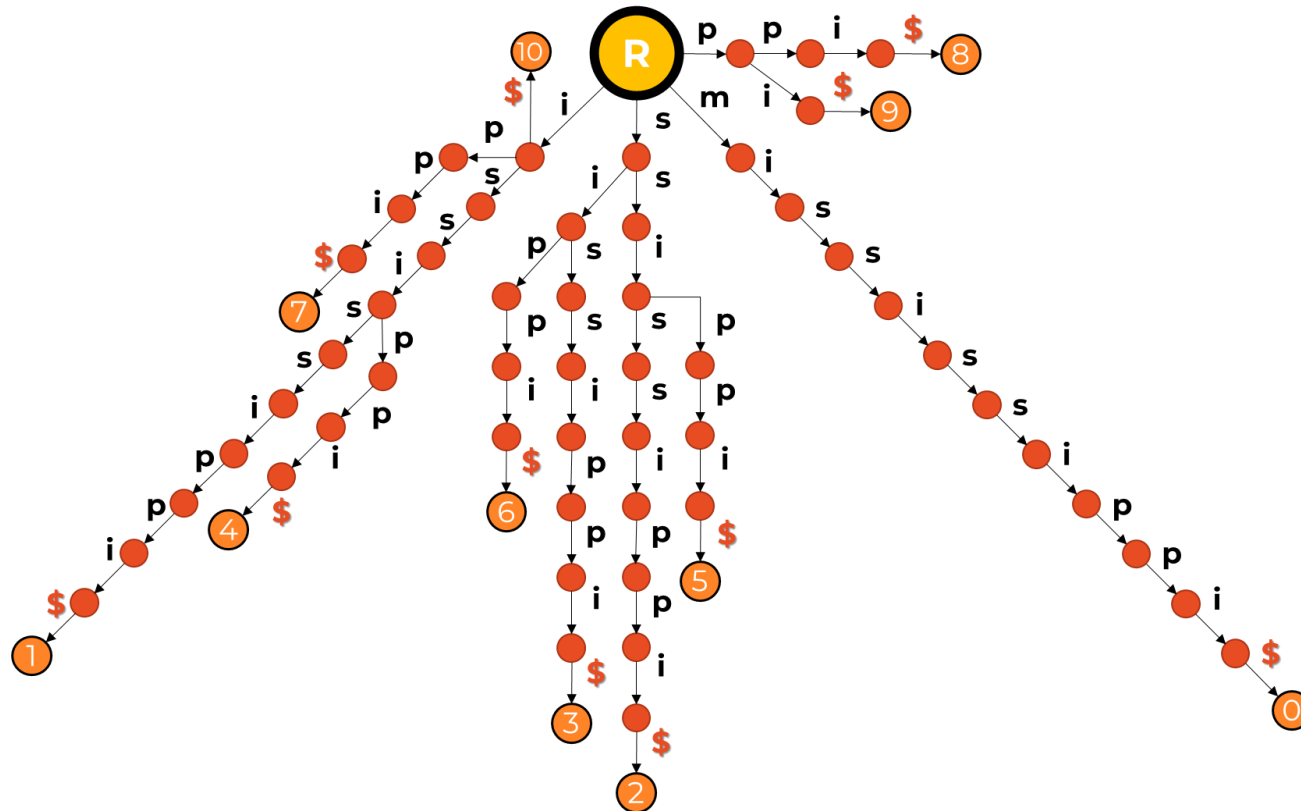
Расход по памяти

$O(N^2)$

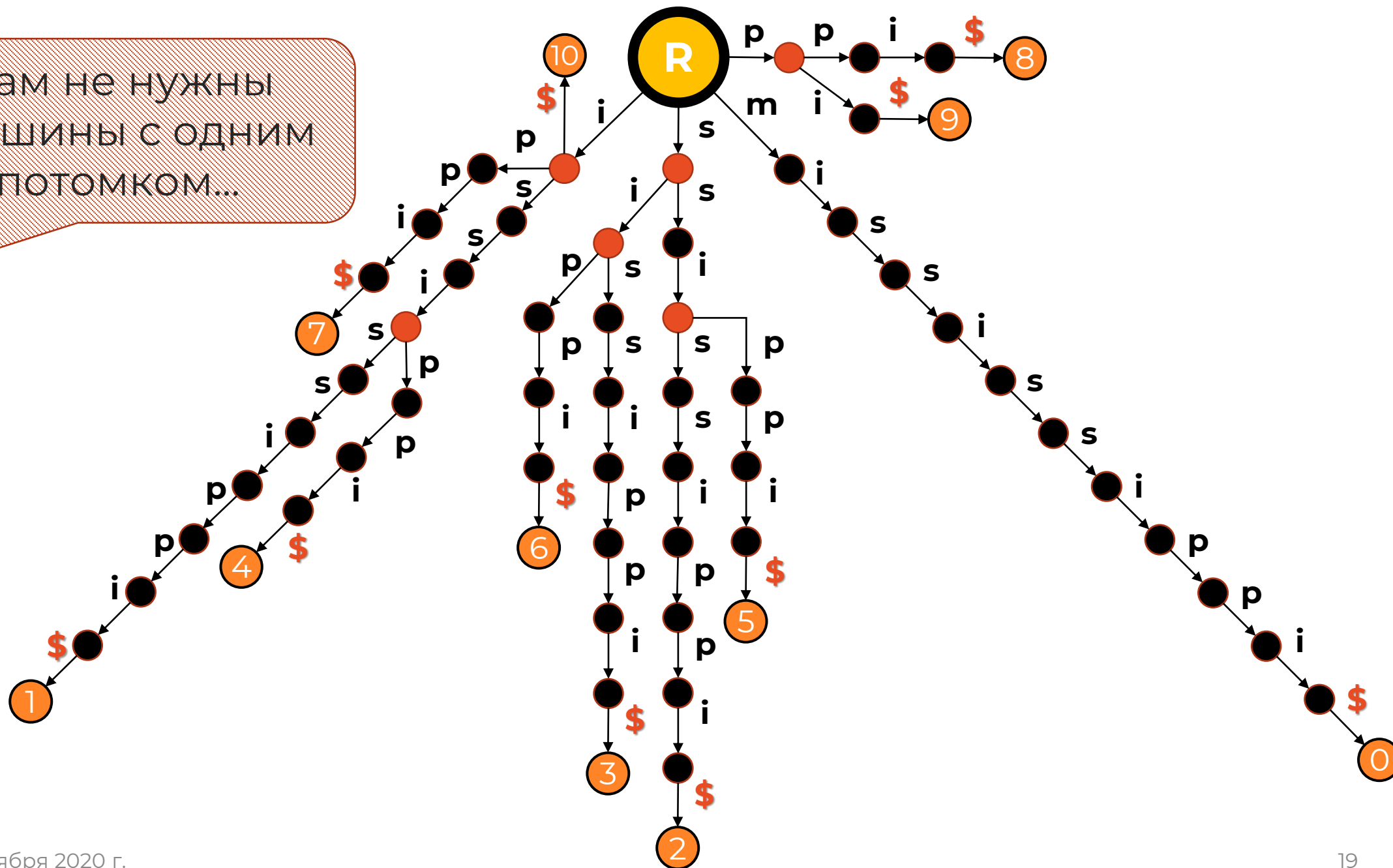
Надо улучшить!

Суффиксное дерево

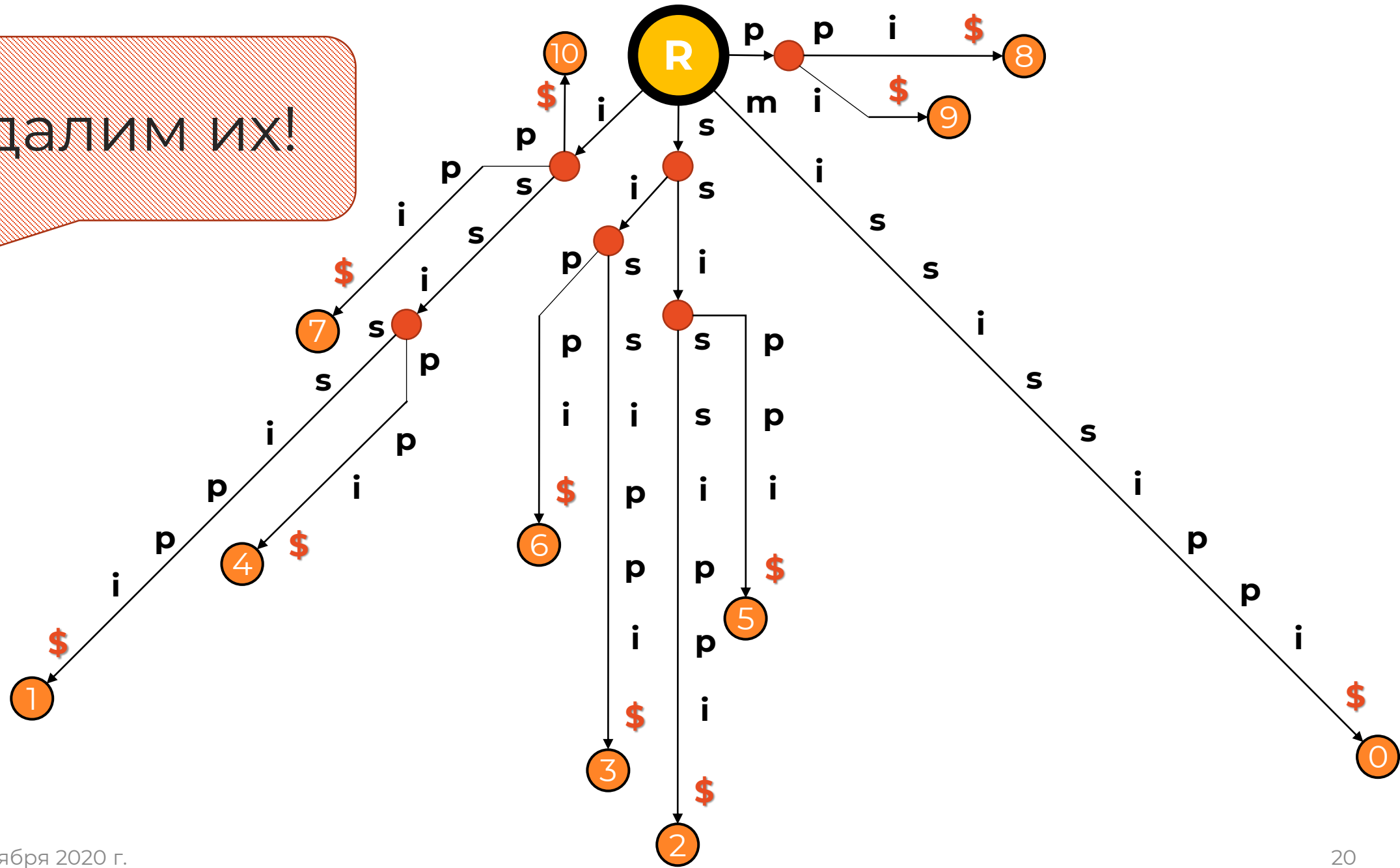
Нам не нужна большая часть вершин бора.



Нам не нужны
вершины с одним
потомком...

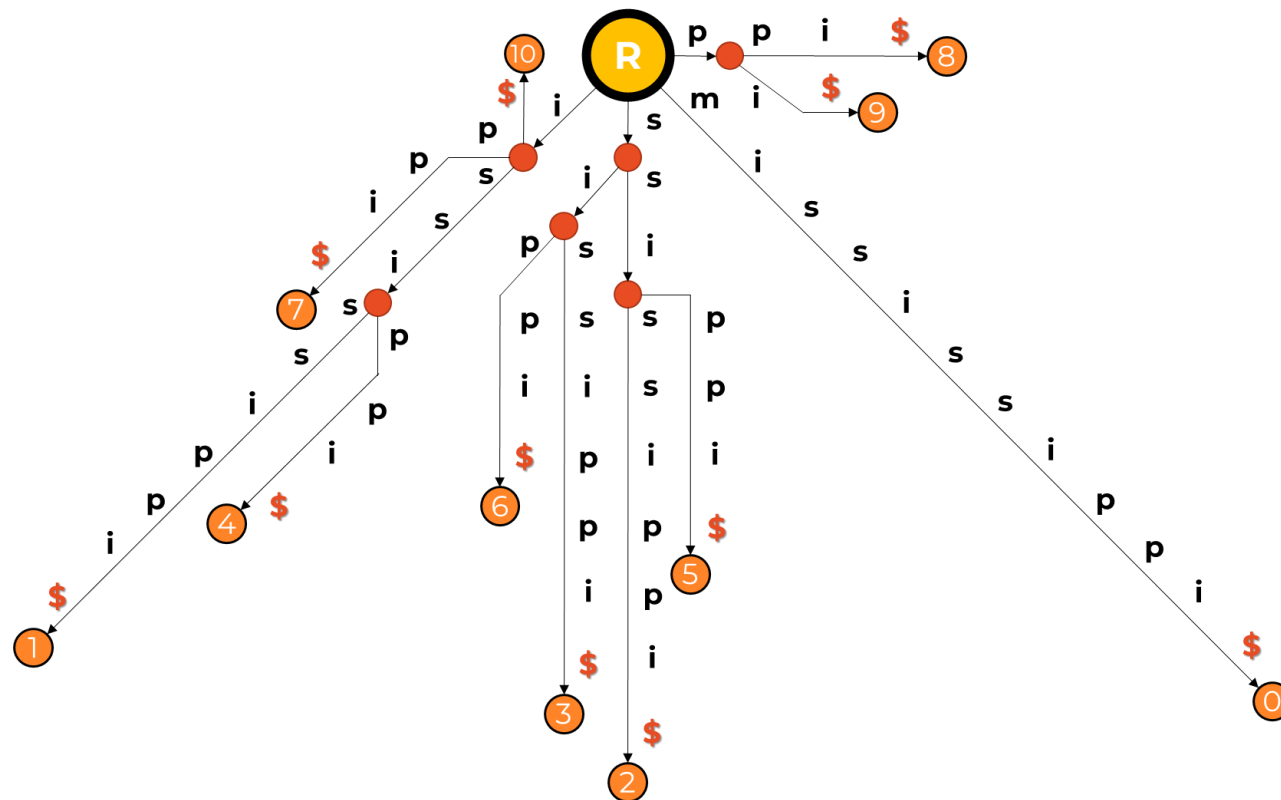


Удалим их!



Теперь на ребрах
бора записаны
подстроки.

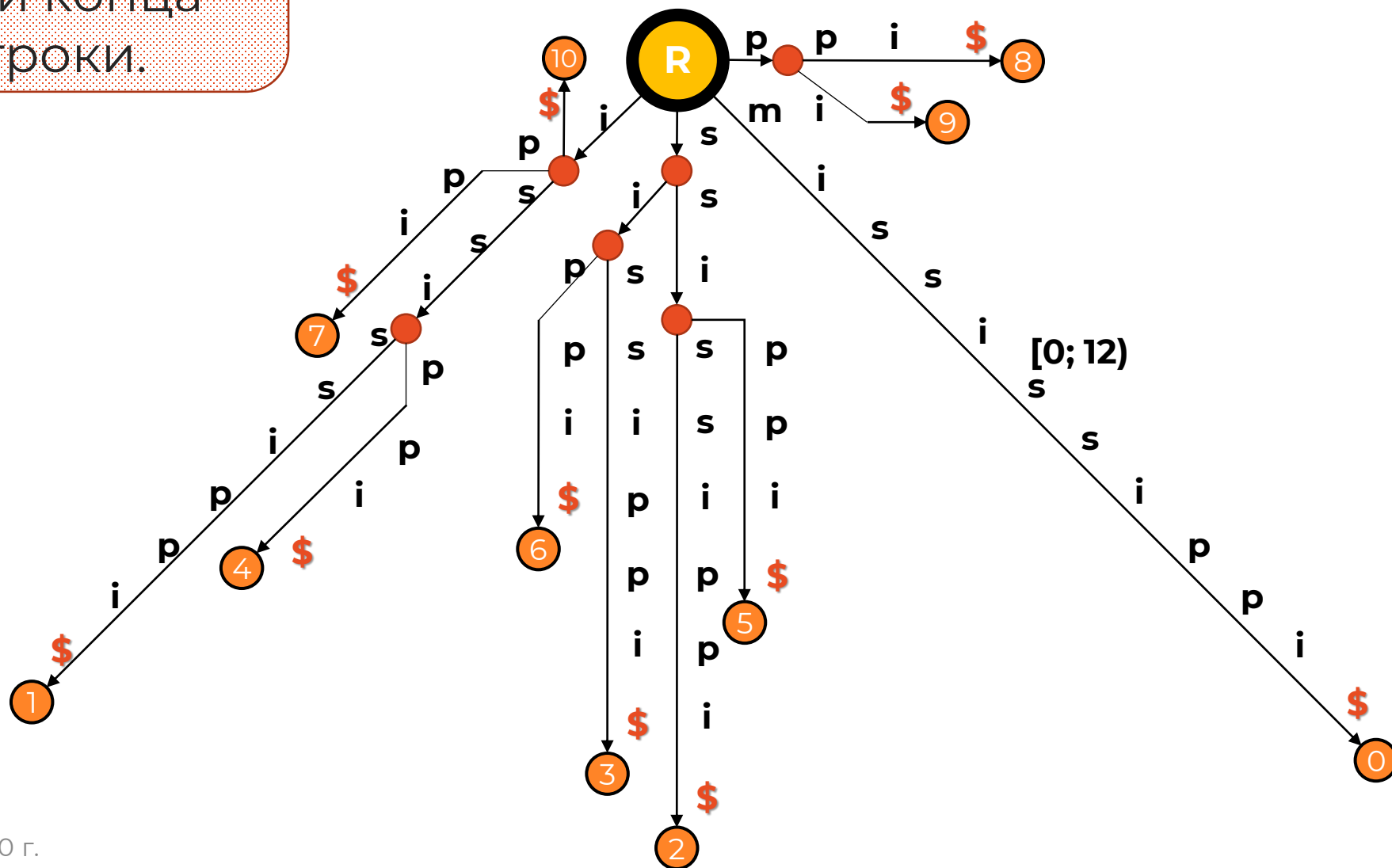
Каков размер
бора?



$O(N^2)$ – мы всё ещё
храним подстроки.

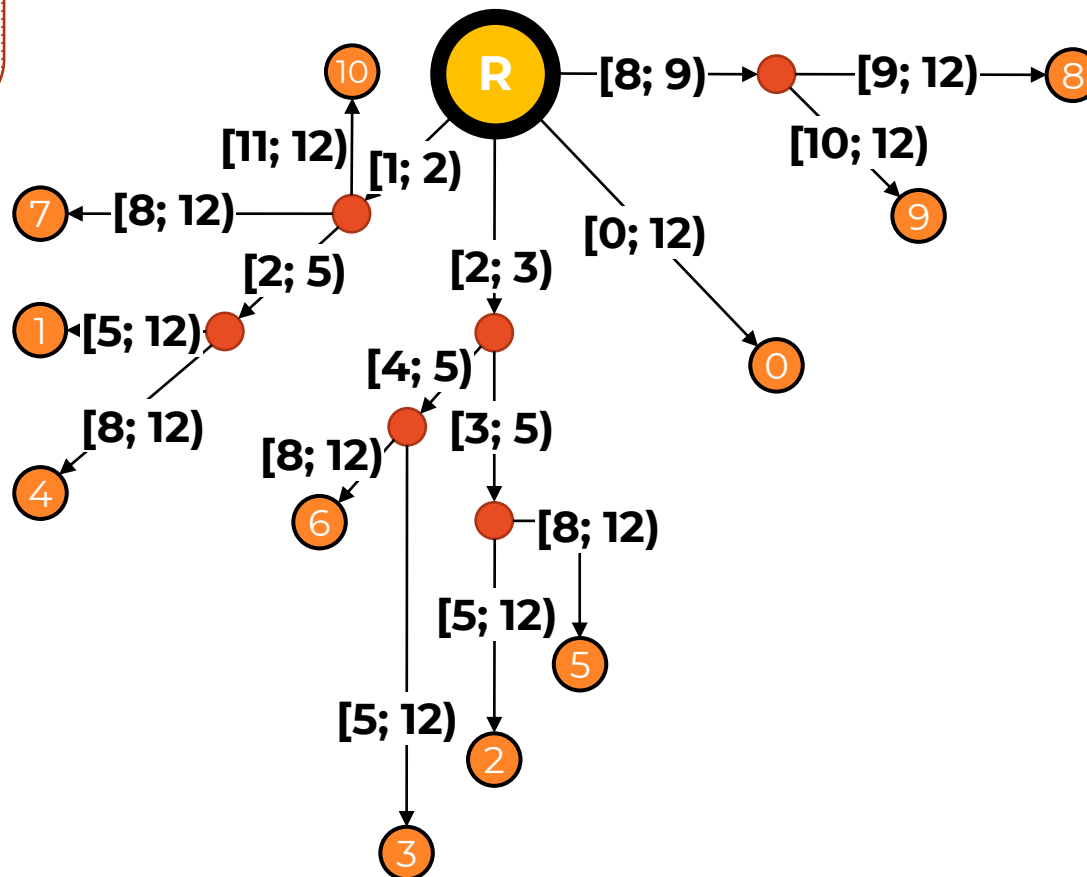
Будем хранить
только позиции
начала и конца
подстроки.

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$



Такая структура
называется
суффиксным
деревом!

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$

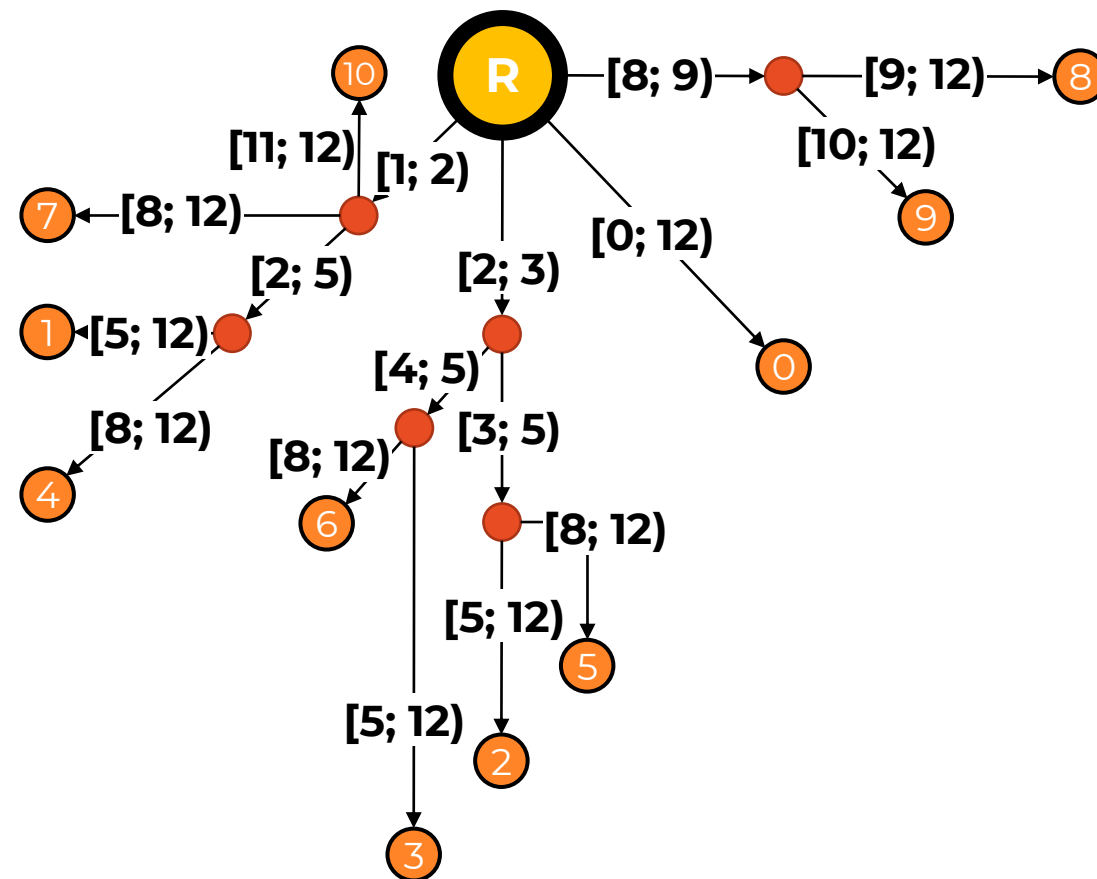


Какой размер у
суффиксного
дерева?

Текст занимает
 $O(N)$.

А всё
остальное?

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$



Сколько вершин у
суффиксного дерева?

?

Пусть листьев **N** . Тогда
не-листьев будет **$N - 1$** .

Всего вершин **$O(N)$** .

Сколько ребер у
суффиксного дерева?

По свойству дерева, на 1
меньше числа узлов.

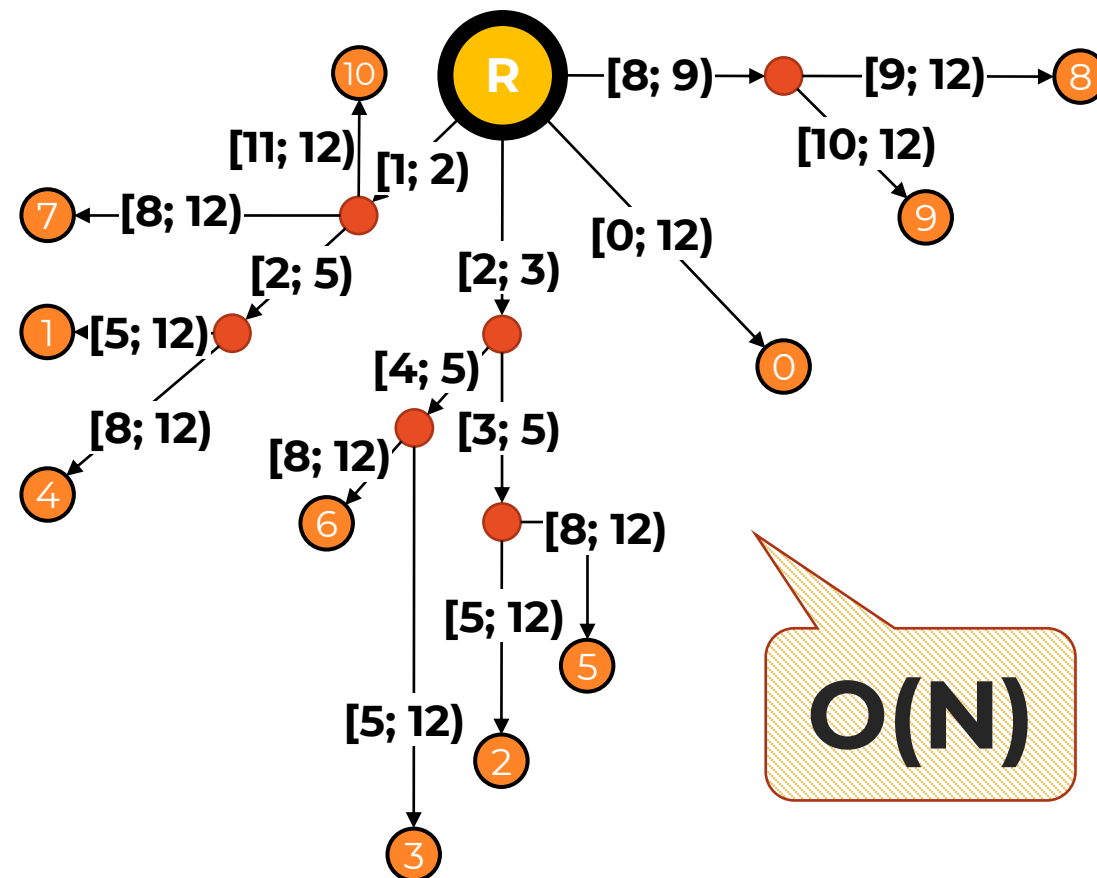
?

Всего **$O(N)$** .

Размер суффиксного дерева в памяти

- $O(N)$ вершин
- $O(N)$ ребер
- $O(N)$ хэш-таблицы в вершинах (суммарно)
- $O(1)$ индексы в ребре
- $O(N)$ занимает паттерн

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$



За какое время **наивно**
строим дерево?

Почему?

$O(N^2)$.

Вначале мы добавляем
суффиксы по одному. Суммарно
размер суффиксов **$O(N^2)$.**

Можно ли
быстрее?

Да, алгоритм
Укконена работает
за **$O(N)$** .

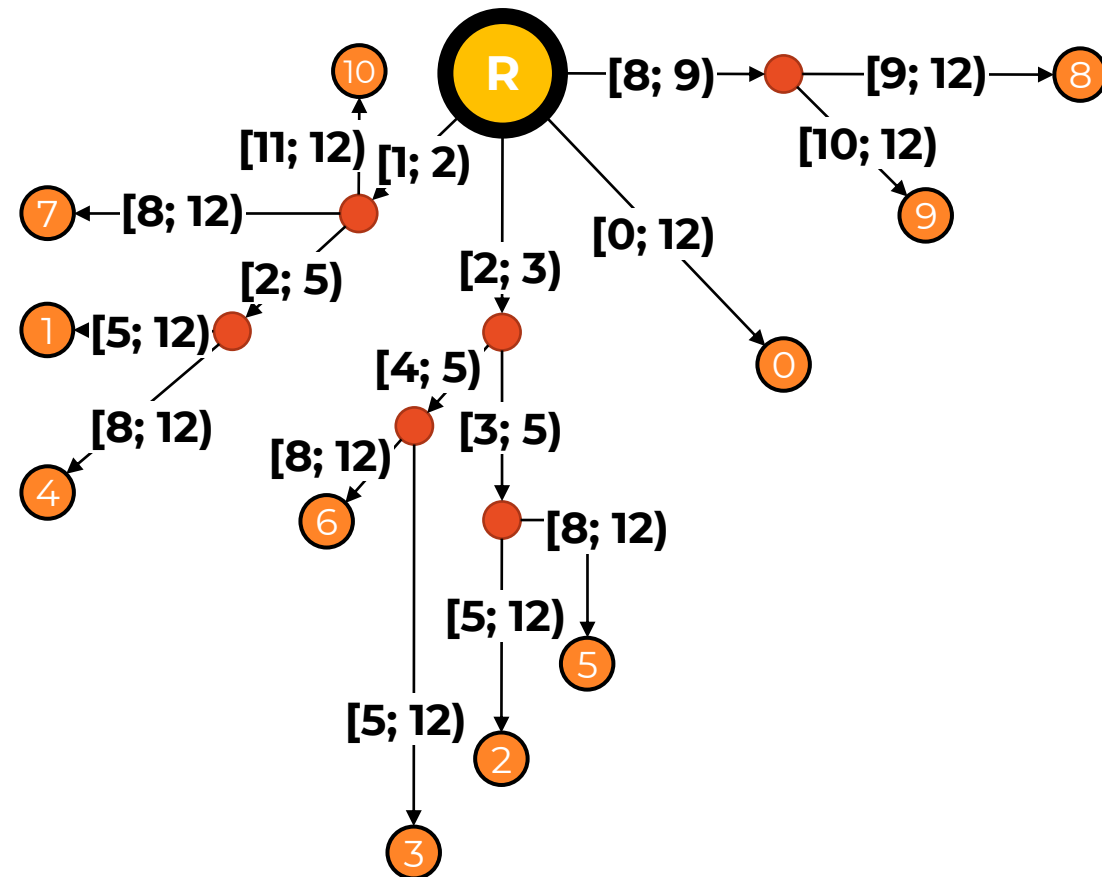
Как искать паттерн?

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$

Ищем паттерн **iss**



- Начинаем в корне;
- Находим ребро, начинающееся с **i**;
- Читаем символы на ребре и сравниваем с паттерном.



Как долго ищем
паттерн?

Читаем паттерн за **$O(M)$** ,
проходим по внутренним
вершинам до листьев за
 $O(M + \alpha)$, где α – число
вхождений.

Какова асимптотика
поиска паттерна с
суффиксным деревом?

Построение дерева

$O(N)$

Поиск паттерна

$O(M + \alpha)$

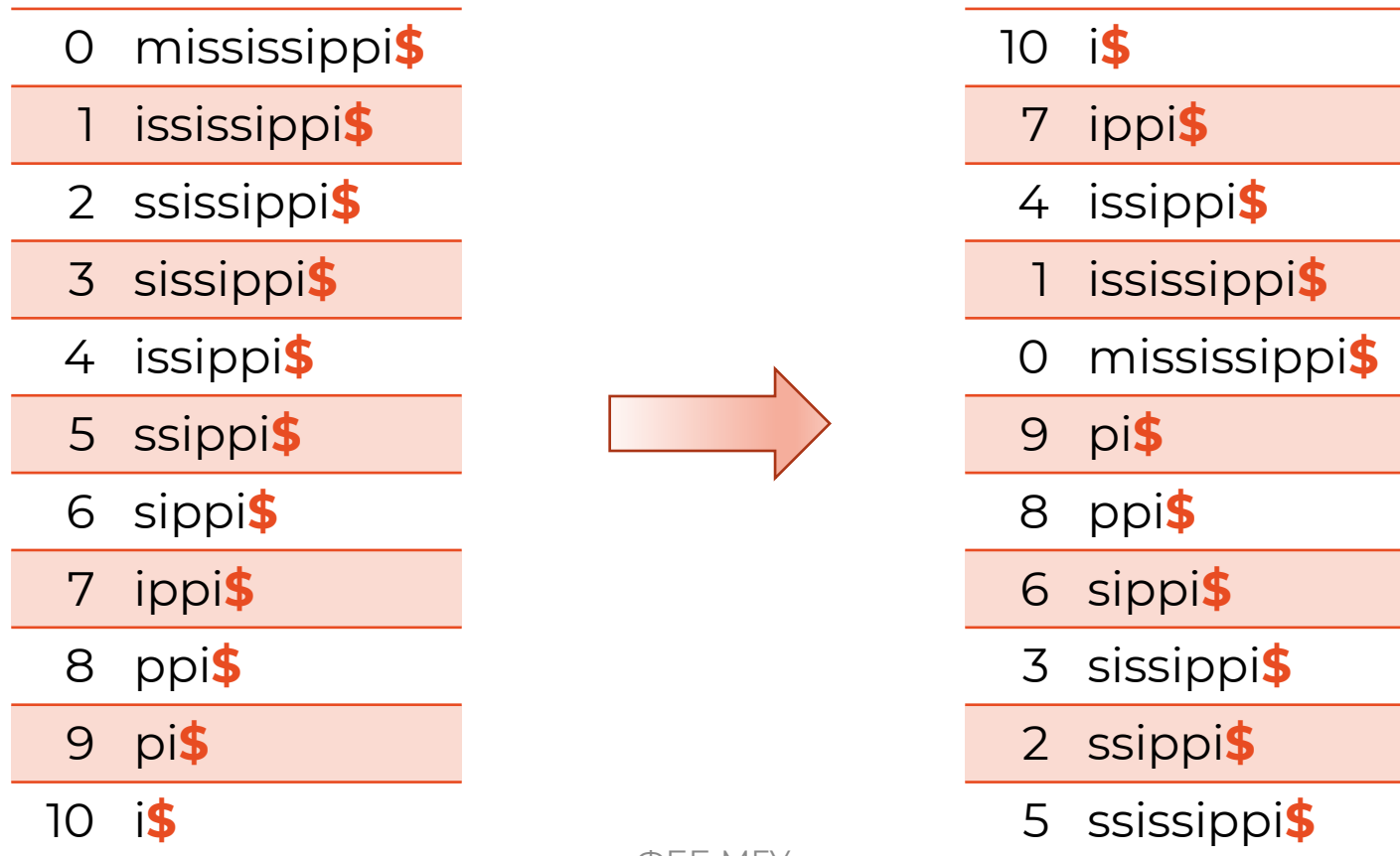
Расход по памяти

$O(N)$

Уже лучше :)

Суффиксный массив

Попробуем отсортировать набор суффиксов:



Суффиксный массив

Какой размер в памяти?

Только если хранить суффиксы явно. Но это необязательно!

$O(N^2)$

10	i\$
7	ippi\$
4	issippi\$
1	ississippi\$
0	mississippi\$
9	pi\$
8	ppi\$
6	sippi\$
3	sissippi\$
2	ssippi\$
5	ssissippi\$

Достаточно хранить
сам паттерн в **$O(N)$**
памяти...

...и позиции суффиксов
в отсортированном
массиве – **$O(N)$** .

0	m
1	i
2	s
3	s
4	i
5	s
6	s
7	i
8	p
9	p
10	i
11	\$

10 i\$

7 ippi\$

4 issippi\$

1 ississippi\$

0 mississippi\$

9 pi\$

8 ppi\$

6 sippi\$

3 sissippi\$

2 ssippi\$

5 ssissippi\$

Достаточно хранить
сам паттерн в **$O(N)$**
памяти...

...и позиции суффиксов
в отсортированном
массиве – **$O(N)$** .

0	m
1	i
2	s
3	s
4	i
5	s
6	s
7	i
8	p
9	p
10	i
11	\$

0	10
1	7
2	4
3	1
4	0
5	9
6	8
7	6
8	3
9	2
10	5

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$

0	1	2	3	4	5	6	7	8	9	10
10	7	4	1	0	9	8	6	3	2	5

Сколько времени
занимает
построение?

Сортируем за **$O(N \log N)$** ;
сравниваем строки за
 $O(N)$.

$O(N^2 \log N)$.

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$

0	1	2	3	4	5	6	7	8	9	10
10	7	4	1	0	9	8	6	3	2	5

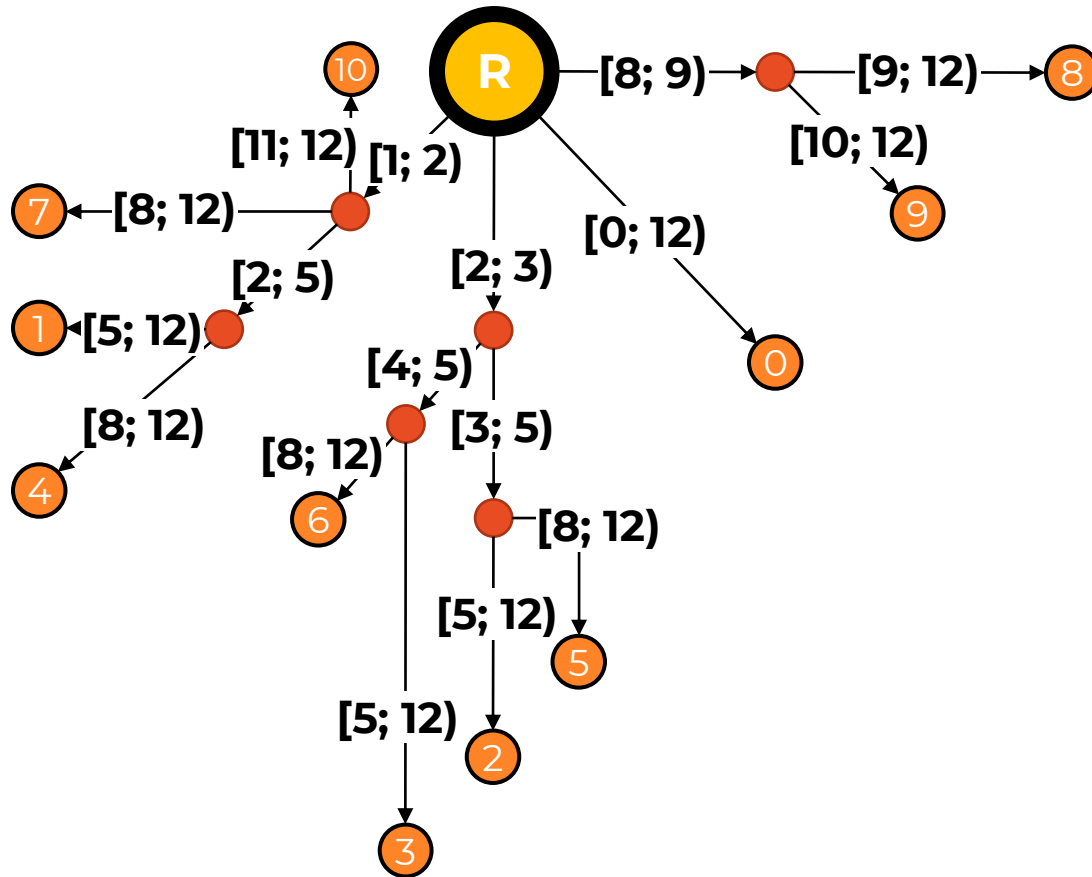
Можно ли строить
быстрее, чем
 $O(N^2 \log N)$?

Существуют алгоритмы
построения за **$O(N \log N)$** ^[1]
и даже за **$O(N)$** ^[2].

За **$O(N)$** также можно
построить из
суффиксного дерева.

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$

0	1	2	3	4	5	6	7	8	9	10
10	7	4	1	0	9	8	6	3	2	5



- Идем из корня;
- На каждом шаге идем в узел, к которому ведет ребро, начинающееся с наименьшего символа.
- Узлы (кроме листьев) кладем в стек.

За какое время
построили
суффиксный массив?

Почти. Обход дерева займет
 $O(N)$, только если мы умеем
сортировать символы в узлах за
 $O(N)$ // они уже отсортированы.
Иначе – **$O(N|\Sigma|\log(|\Sigma|))$** .

$O(N)$?

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$

0	1	2	3	4	5	6	7	8	9	10
10	7	4	1	0	9	8	6	3	2	5

Как искать в
отсортированном
массиве?

Бинарным поиском!

Ищем паттерн **iss**

- Вначале ищем левую границу вхождений;

Что делаем после этого?

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$

i	s	s
---	---	---



0	1	2	3	4	5	6	7	8	9	10
10	7	4	1	0	9	8	6	3	2	5

Нет! Это приведет
к времени
порядка **$O(MN)$** .

Нужно провести
бинарный поиск с
условием $<$ вместо $\leq$

Можно
последовательно
сравнить суффиксы
от 3 до 10 позиций?

Также нужно искать
средний элемент по
формуле $(L+R+1)/2$
вместо $(L+R)/2$.

Ищем паттерн **iss**

- Ищем правую границу вхождений.

iss ВХОДИТ В СТРОКУ
В ПОЗИЦИЯХ **4** И **1**.

0	1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i	\$



A diagram illustrating a 2D array structure. The array is represented as a grid of 11 columns and 2 rows. The top row contains indices 0 through 10. The bottom row contains values 10, 7, 4, 1, 0, 9, 8, 6, 3, 2, 5. Above the grid, three arrows indicate pointers: a red arrow labeled 'L' points to the first column (index 0), a purple arrow points to the sixth column (index 5), and an orange arrow labeled 'R' points to the last column (index 10). The cell at row 1, column 2 (value 4) is highlighted in orange, matching the orange arrow 'R'.

За какое время ищем в
суффиксном массиве?

Да, но сравнение
строк занимает **$O(M)$** ,
поэтому нам нужно
 $O(M \log N)$ времени.

$O(\log N)$?

Как думаете, можно ли быстрее?

Да. Дополнительным препроцессингом можно улучшить сначала средний случай, а потом добиться асимптотики **$O(\log N + M)$** .



habr.com/ru/post/115346/

Какова асимптотика
поиска паттерна с
суффиксным массивом?

Построение массива

$O(N)$

Поиск паттерна

$O(\log N + M)$ [+ $O(\alpha)$]

Расход по памяти

$O(N)$

Замечательно...



Спасибо за внимание!