



УНИВЕРСИТЕТ ИТМО

Лекция 1

Поиск подстрок

Алгоритмы **поиска подстроки** – одна из базовых задач поиска информации, заключающаяся в поиске определенного шаблона в строке.

Шаблон (needle, иголка) – конструкция, заданная в помощью определенного алфавита Σ .

Строка (haystack, стог сена) – исходный текст, в котором ведется поиск.

Шаблон: Грека

*Строка: Ехал Грека через реку, Видит Грека – в реке рак.
Сунул Грека руку в реку – Рак за руку*

*Результат: Ехал **Грека** через реку, Видит **Грека** – в реке рак.
Сунул **Грека** руку в реку – Рак за руку*

или

Количество совпадений: 3

Наивный алгоритм

Наивный алгоритм (прямой алгоритм, примитивный алгоритм, brute force algorithm) – прямое последовательное сравнение шаблона с элементами строки.

Шаблон: 1 0 1

Строка: 1 1 1 0 3 8 4 7 1 8 1 0 1 3 8 0

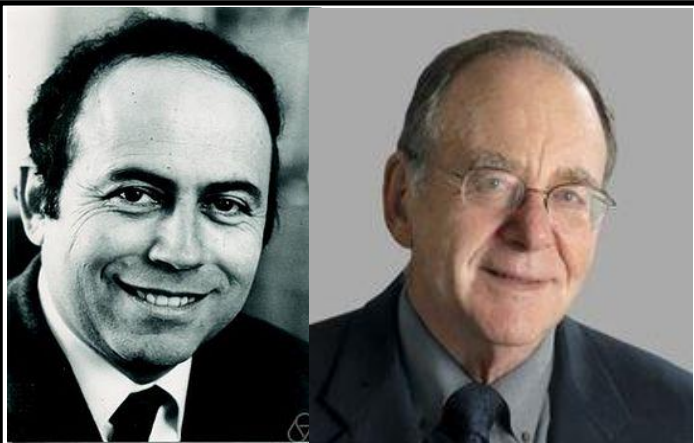
Шаг 1 – сравнение символов Строки и Шаблона по элементам:

1 1 1 0 3 8 4 7 1 8 1 0 1 3 8 0
↑ ↑ ↑
1 0 1

Если элементы совпали – конец работы. Если элементы не совпали:

Шаг 2 – сдвиг шаблона на одну позицию и повторное Шага 1:

1 1 1 0 3 8 4 7 1 8 1 0 1 3 8 0
↑ ↑ ↑
1 0 1



М. О. Рабин

Р. М. Карп

Алгоритм Рабина-Карпа

Алгоритм Рабина-Карпа – алгоритм, который ускорить работу наивного алгоритма за счет использования хэш-функции.

Шаблон: *ТОТ*

Строка: *ЭТОИЭТОТ*

Шаг 1 – Вычисляем размер алфавита, длину шаблона и назначаем числовые значения для символов:

*Т, О, Э, И – количество разных символов. Размер алфавита $x = 4$.
Длина шаблона $m=2$ (через индексы). $T=0$, $O=1$, $E=2$, $I=3$*

Шаг 2 – Производим расчет хэша для шаблона:

$$H = n_0 \cdot x^m + n_1 \cdot x^{m-1} + \dots + n_m \cdot x^0$$

$$H(TOT) = 0 \cdot 4^2 + 1 \cdot 4^1 + 0 \cdot 4^0 = 4$$



Шаг 3 – Производим расчет хэша для элементов строки, равных длине шаблона:

ЭТОИЭТОТ

$H(\text{ЭТО}) =$

$$= 2 * 4^2 + 0 * 4^1 + 1 * 4^0 = 33$$

ЭТОИЭТОТ

$H(\text{ТОИ}) =$

$$= 0 * 4^2 + 1 * 4^1 + 3 * 4^0 = 33$$

ЭТОИЭТОТ

$H(\text{ОИЭ}) =$

$$= 1 * 4^2 + 3 * 4^1 + 1 * 4^0 = 33$$

ЭТОИЭТОТ

$H(\text{ИЭТ}) =$

$$= 3 * 4^2 + 2 * 4^1 + 0 * 4^0 = 33$$

ЭТОИЭТОТ

$H(\text{ЭТО}) =$

$$= 2 * 4^2 + 0 * 4^1 + 1 * 4^0 = 33$$

ЭТОИЭТОТ

$H(\text{ТОТ}) =$

$$= 0 * 4^2 + 1 * 4^1 + 0 * 4^0 = 33$$

Шаг 4 – Поочередное сравнение хэша шаблона и элементов строк

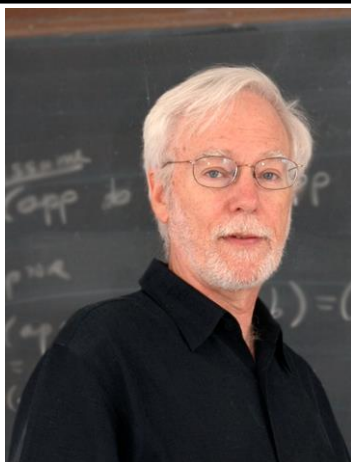
$$H(\text{ЭТОТ}) \neq H(\text{ТОТ}) \rightarrow H(\text{ТОИ}) \neq H(\text{ТОТ}) \rightarrow \dots \rightarrow H(\text{ТОТ}) = H(\text{ТОТ})$$

Если хэш совпал, то происходит Шаг 1 алгоритма наивного поиска

Если Шаг 1 алгоритма наивного поиска не совпал, возвращение к Шагу 4



Р. С. Бойер



Д. С. Мур

Алгоритм Бойера-Мура

Алгоритм Рабина-Карпа – алгоритм, который сдвигает шаблон или до первого совпадающего символа, или на длину шаблона

Шаблон: **Т О Т**

Строка: **Э Т О И Э Т О Т**

Шаг 1 – Рекурсивно сравниваем первые элементы строки и шаблона

Э Т О И Э Т О Т
 ↑ ↑ ↑
Т О Т

Шаг а – сравнение «О» и «Т»

Шаг б – сравнение «Т» и «О»

Шаг в – сравнение «Э» и «Т»

Шаг 2.1 – Поскольку на шаге а получили «ложь», проверяем (справа-налево), есть ли буква «О» на другой позиции шаблона, если есть, то сдвигаем до совпадающего значения и повторяем Шаг 1.

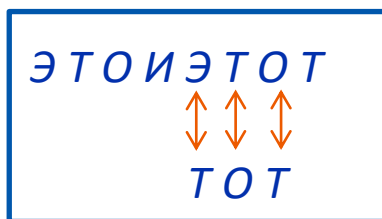
Э Т О И Э Т О Т
 ↑ ↑ ↑
Т О Т

Шаг а – сравнение «И» и «Т»

Шаг б – сравнение «О» и «О»

Шаг в – сравнение «Т» и «Т»

Шаг 2.2 – Поскольку на шаге а получили «ложь», проверяем (справа-налево), есть ли буква «И» на другой позиции шаблона, если нет, то сдвигаем шаблон на его длину и повторяем Шаг 1.

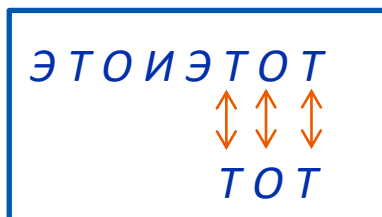


Шаг а – сравнение «О» и «Т»

Шаг б – сравнение «Т» и «О»

Шаг в – сравнение «Э» и «Т»

Происходит повтор ситуации из шага 2.1. Сдвигаем шаблон, чтобы сопоставить первую букву «О» шаблона с буквой «О» строки:



Шаг а – сравнение «Т» и «Т»

Шаг б – сравнение «О» и «О»

Шаг в – сравнение «Т» и «Т»



Возвращаемся к шагу 1.

Совпало, шаблон в строке найден.



Префикс и суффикс

Префикс – подстрока, начинающаяся с первого элемента строки.

Суффикс – подстрока, заканчивающаяся последним элементом строки.

Строка:

Строка которую мы рассматриваем

Префиксы:

С
Ст
Стр
Стро
Строк
Строка
Строкак
Строкако
...

Суффиксы:

м
ем
аем
ваем
иваем
риваем
триваем
атриваем
...



Префикс-функция

Префикс-функция (π) — максимальная длина совпадающих суффиксов и **префиксов**, для каждого префикса строки, не равных длине самой подстроки (самого префикса).

Строка:

abaabaabbab

$\pi(\text{строки}) = [0, 0, 1, 1, \dots, 2]$

Префикс 1: a

$\pi(a) = 0$ (нет суффиксов/префиксов не равных длине префикса 1)

Префикс 2: ab

$\pi(ab) = 0$ (сравнение суффиксов/префиксов: «a» и «b»)

Префикс 3: aba

$\pi(a) = 1$ (совпадают префикс «a» и суффикс «a» длиной 1 (не совпали «ab» и «ba»))

Префикс 3: abaa

$\pi(a) = 1$ (совпадают префикс «a» и суффикс «a» длиной 1 (не совпали «ab» и «aa», «aba» и «baa»))

...

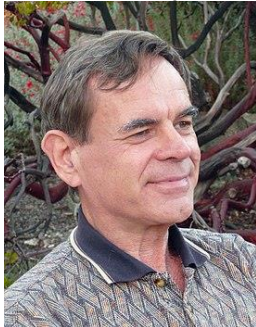
Префикс 11: abaabaabbab

$\pi(a) = 2$ (совпадают префикс «ab» и суффикс «ab» длиной 2 (не совпали «a» и «b», «aba» и «bab», «abaa» и «bbab», «abaab» и «abbab» и т.д.))

Алгоритм Кнута-Морриса-Пратта



Д. Э. Кнут



В. Р. Пратт



Д. Х. Моррис

Алгоритм Кнута-Морриса-Пратта (КМП) – эффективный алгоритм, время работы которого линейно зависит от входных данных.

Шаблон: *Т О Т*

Строка: *Э Т О И Э Т О Т*

Шаг 1 – поиск префикс-функции для шаблона:

$\pi(T)=0$

$\pi(ТО)=0$

$\pi(ТОТ)=1$



$\pi(\text{шаблона})=[0, 0, 1]$

Шаг 2 – сравнение элементов строки и шаблона (слева-направо):

Э Т О И Э Т О Т

Т О Т

Если символы совпали, то шаблон в строке найден.



Шаг 3 – сдвигаем по формуле: длина совпавшего участка - значение префикс функции для не совпавшего элемента + 1

Длина (слева-направо) совпавшего участка : 0

Значение префикс функции для не совпавшего эл-та (T): 0

Сдвиг: $0 - 0 + 1$

Результат:

Э Т О И Э Т О Т
↑ ↑ ↑
Т О Т

Возвращаемся к шагу 2.

Обнаруживается несовпадение, очередной переход к шагу 3

Длина (слева-направо) совпавшего участка : 2

Значение префикс функции для не совпавшего эл-та (TOT): 1

Сдвиг: $2 - 1 + 1$

Результат:

Э Т О И Э Т О Т
↑ ↑ ↑
Т О Т

Возвращаемся к шагу 2.


Обнаруживается несовпадение, очередной переход к шагу 3

Длина (слева-направо) совпавшего участка : 0

Значение префикс функции для не совпавшего эл-та (T): 0

Сдвиг: $0 - 0 + 1$

Результат:

Э Т О И Э Т О Т

 Т О Т

Возвращаемся к шагу 2.

Обнаруживается несовпадение, очередной переход к шагу 3

Длина (слева-направо) совпавшего участка : 0

Значение префикс функции для не совпавшего эл-та (T): 0

Сдвиг: $0 - 0 + 1$

Результат:

Э Т О И Э Т О Т

 Т О Т

Возвращаемся к шагу 2. Совпало, шаблон в строке найден.