

Алгоритмы и структуры данных

Алгоритмы на строках. Часть II Месть Кнута-Морриса-Пратта

Кухтичев Антон



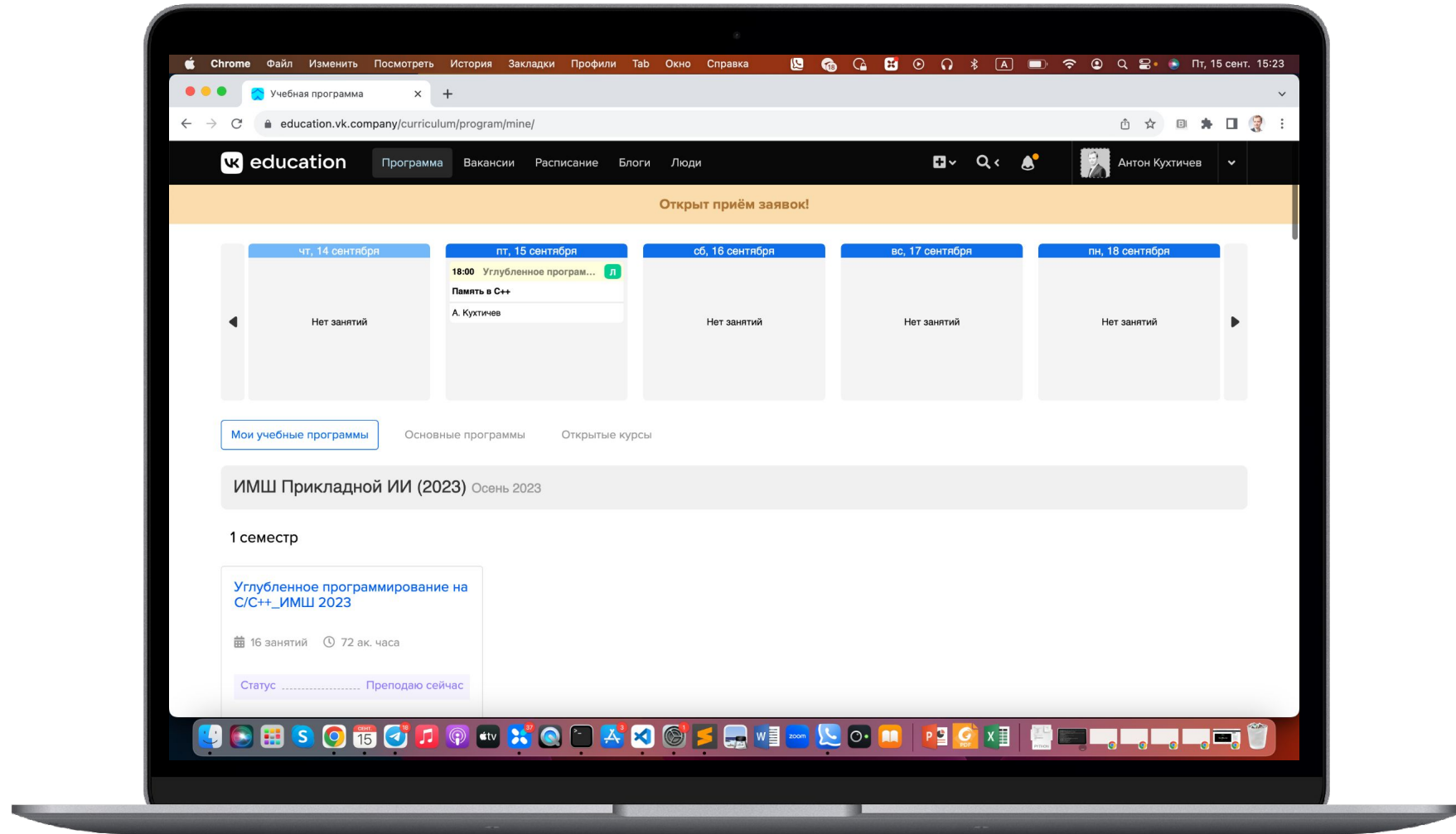
12 мая 2025 года

Содержание занятия

- Рабин-Карп
- Кнут-Моррис-Пратт (КМП)
- Алгоритм Ахо-Корасика

Напоминание отметиться на портале

и оставить отзыв
после лекции



Литература



Строки, деревья и последовательности в алгоритмах -
Гасфилд Д.М.

Алгоритм Рабина-Карпа



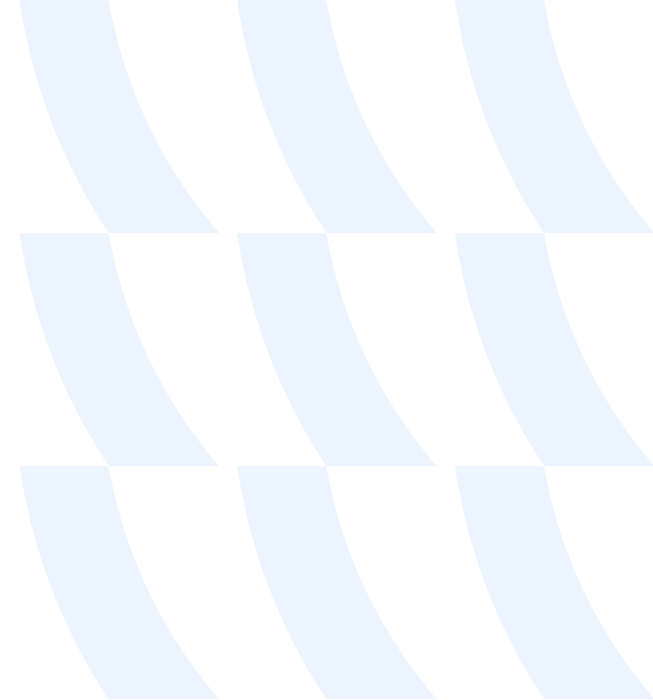
Идея

- Дан текст T длины n и строка P длины m ;
- Улучшение наивного алгоритма
- Сравниваем не символы, а хеши!

012345678901

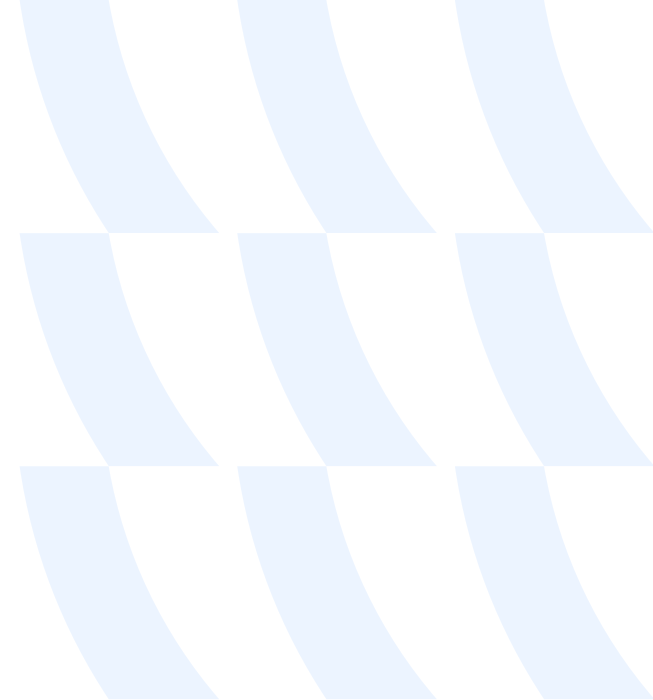
$T = \text{aabcaabxaz}$ $P_0 = \text{hash}(\text{aab})$, $P_1 = \text{hash}(\text{abc})$, $P_2 = \text{hash}(\text{bca})$

$P = \text{aab}$ $P_h = \text{hash}(\text{aab})$



Наивный алгоритм

```
rabin_karp(T, P):  
    n = length(T)  
    m = length(P)  
    hp = hash(P[1..n])  
    for i in 1..n-m+1 do  
        ht = hash(T[i..i+m])  
        if ht == hp then  
            if T[i..i+m] = P then  
                return i  
            fi  
        fi  
    done
```



Упражнение #2

Реализовать простейший вариант поиска подстроки в строке при помощи алгоритма Рабина-Карпа

```
def rapin_karp(text: str, pattern: str) -> bool:  
    pass
```

<https://interview.cups.online/live-coding/?room=ee2ffa07-3c74-417e-9540-257c1782c1d3>

Кнут-Морисс-Пратт



Идея

Для каждой позиции i образца P определим $sp_i(P)$ как длину наибольшего собственного суффикса $P[1..i]$, который совпадает с префиксом P .

12345678901

$P = \text{abcaeabcabd}$

$sp_2 = sp_3 = 0$ abcaeabcabd

$sp_4 = 1$ $\text{abca} \text{eabcabd}$

$sp_8 = 3$ $\text{abcae} \text{abcabd}$

$sp_{10} = 2$ $\text{abcaeabc} \text{abd}$

Препроцессинг

- Нужно за линейное время определить все sp_i
- Позиция $j > 1$ отображается в i , если $i = j + Z_j(P) - 1$. То есть i - это правый конец Z-блока, начинающего в j .

```
for i := 1 to n do
    spi := 0
for j := n downto 2 do begin
    i := j + Z_j(P) - 1
    sp_i := Z_j
end
```

Алгоритм КМП

begin

Обработать P , найдя $F'(k) = sp'_{k-1} + 1$ для k от 1 до $n + 1$.

$c := 1$

$p := 1$

while $c + (n - p) < m$ do begin

 while $P(p) = T(c)$ и $p \leq n$ do begin

$p := p + 1$

$c := c + 1$

 end

 if $p == n + 1$ then

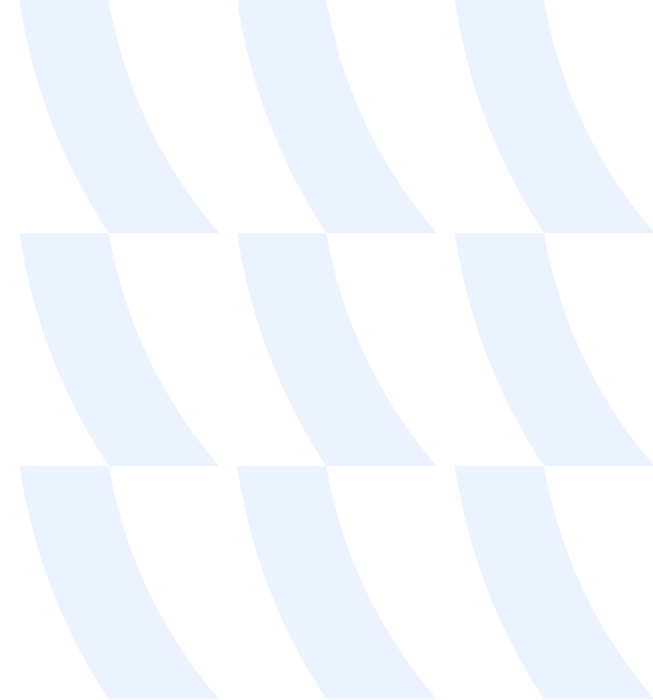
 зафиксировать вхождение P в T , начиная с позиции $c - n$.

 if $p == 1$ then $c := c + 1$

$p := F'(P)$

 end

end



Упражнение #3

Реализовать поиск подстроки в строке при помощи алгоритма Кнута-Морриса-Пратта

```
def KnuthMorrisPratt(text: str, pattern: str) -> bool:  
    pass
```

<https://interview.cups.online/live-coding/?room=ced41f5b-db37-4765-8546-d8e588d1d9ab>

Ахо-Корасик



Идея

- Задано множество образцов $P = \{P_1, P_2, \dots, P_n\}$, требуется обнаружить вхождения в тексте T всех образцов.
- Если взять алгоритм поиска за линейное время, то сложность будет $O(n+zm)$
- Альфред Ахо и Маргарет Корасик предложили алгоритм поиска за $O(n + m + k)$, k – число вхождений в T образцов из P .

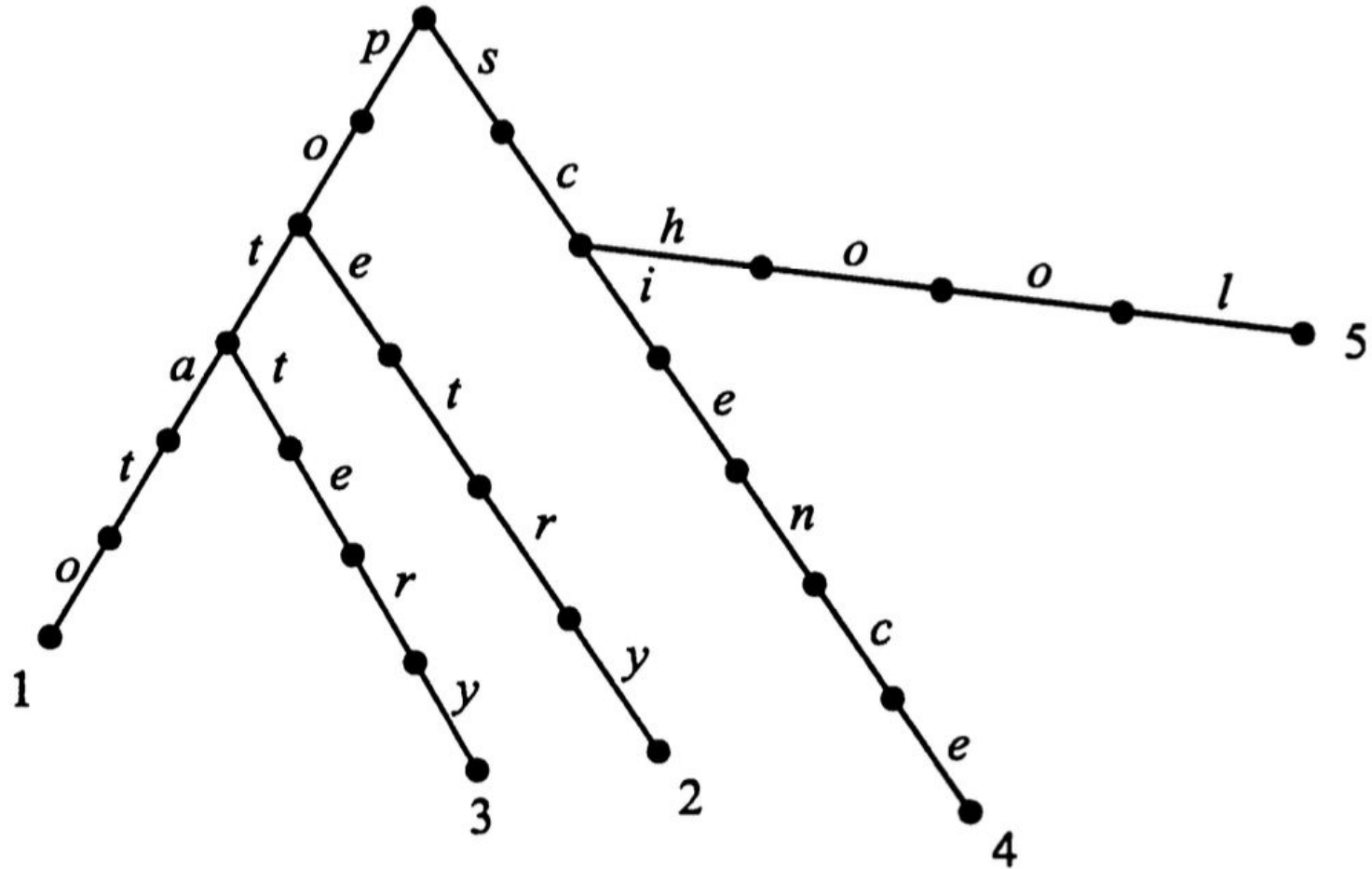
Дерево ключей

Деревом ключей (keyword tree) для множества P называется ориентированное дерево с корнем K , удовлетворяющее трем условиям:

1. каждая дуга помечена ровно одним символом;
2. любые две дуги, выходящие из одной и той же вершины, имеют разные пометки;
3. каждый образец P_i , в P отображается в некоторую вершину v из K , такую что символы на пути из корня K в v в точности составляют P_i , и каждый лист из K соответствует какому-либо образцу из P .

Дерево ключей

Дерево ключей для множества образцов {potato, poetry, pottery, science, school}

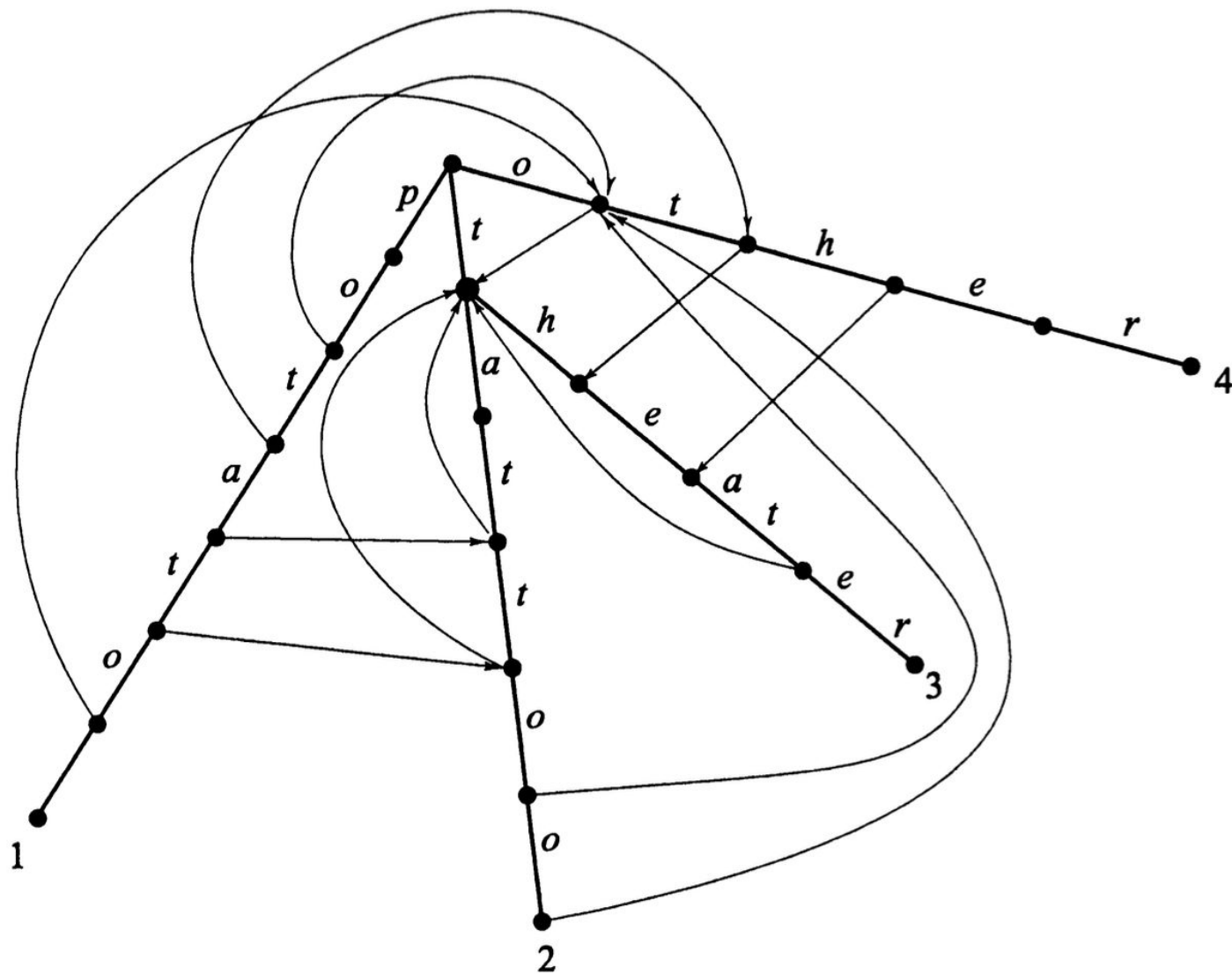


Функция неудач

Каждая вершина v в K помечена строкой, полученной конкатенацией символов на пути от корня K до вершины и в порядке их появления. Для этой пометки используется обозначение $L(v)$. Так что конкатенация символов пути от корня до v произносит строку $L(v)$.

Для любой вершины v дерева K определим $l_p(v)$ как длину наибольшего собственного суффикса строки $L(v)$, которая является префиксом некоторого образца из P .

Функции неудач



Алгоритм Ахо-Корасика

```
l := 1;
c := 1;
w := корень K
repeat
    while есть дуга (w, w'), помеченная символом T(c) begin
        if w' занумерована образцом i или существует путь из связей
неудач из w' в вершину с номером i; then
            сообщить, что Pi встретилось в T, начиная с позиции l;
            w := w';
            c := c + 1;
        end;
    w := nw
    l := c - lp(w)
until c > n;
```

Домашнее задание



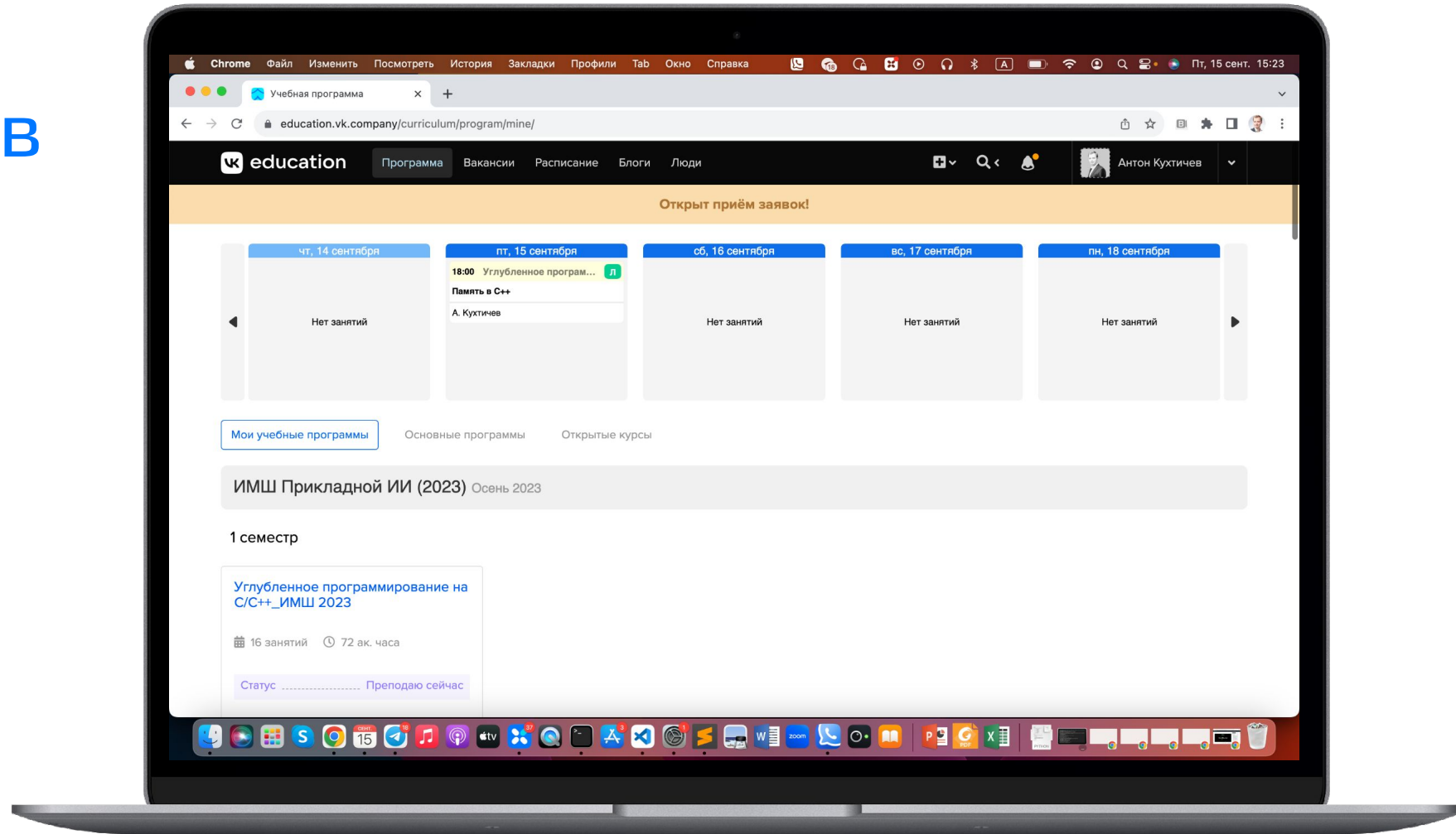
Домашнее задание

Необходимо реализовать алгоритм Ахо-Корасика для поиска образцов в тексте для указанного алфавита.

Вариант алфавита: Слова не более 16 знаков латинского алфавита (регистронезависимые).

Напоминание оставить отзыв

Это правда важно





Спасибо
за внимание!