

Алгоритмы и структуры данных

Алгоритмы на строках. Часть I Новая надежда

Кухтичев Антон



education

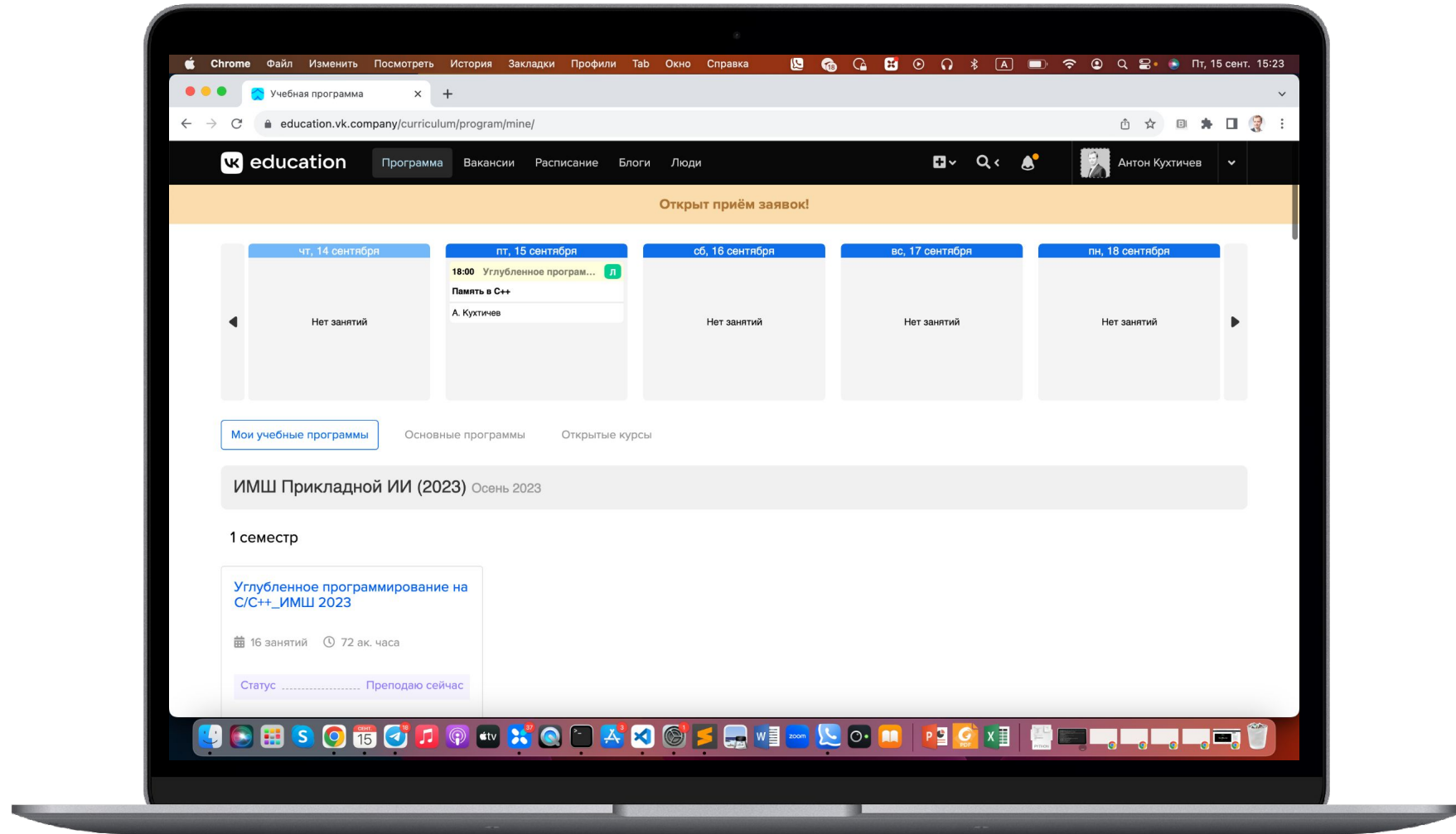
19 апреля 2024 года

Содержание занятия

- Квиз
- Наивный алгоритм
- Z-алгоритм
- Бойер-Мур

Напоминание отметиться на портале

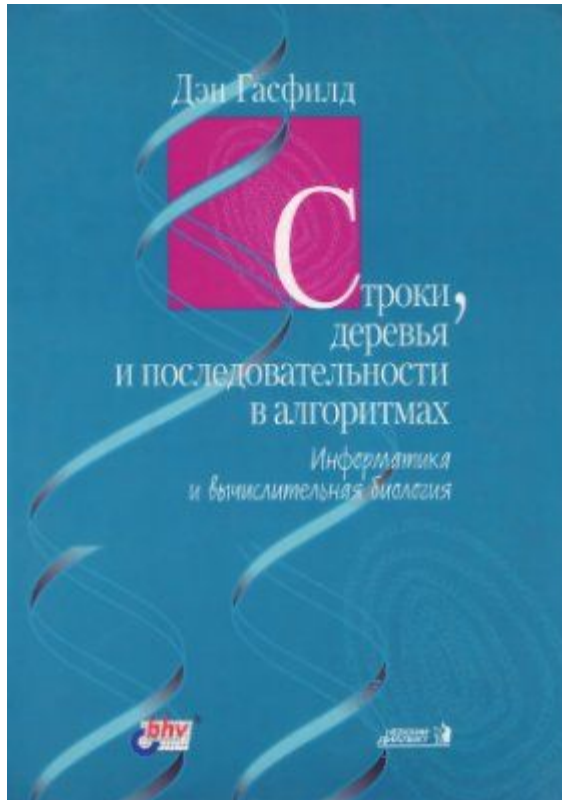
и оставить отзыв
после лекции



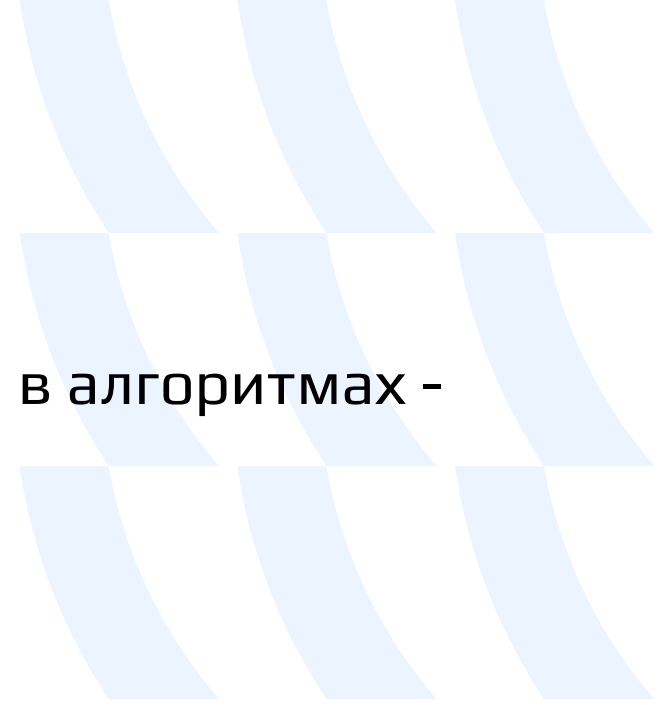
КВИЗ



Литература



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



Наивный алгоритм

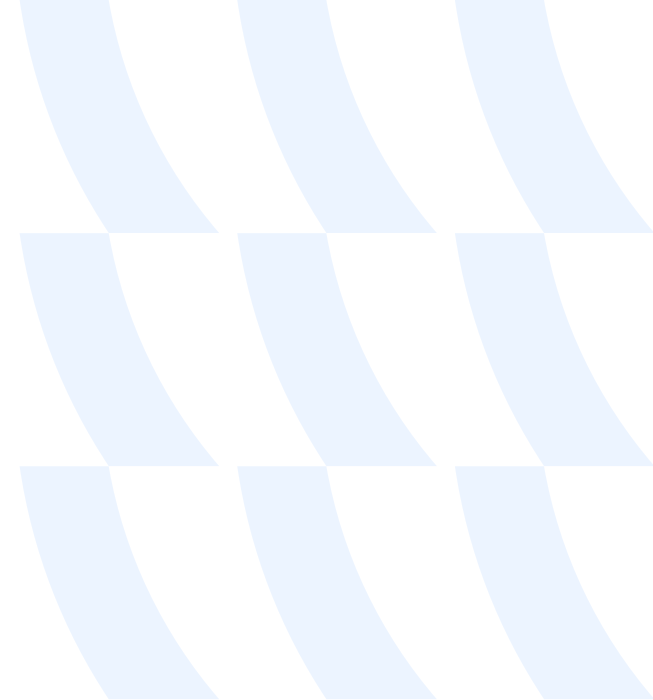


Наивный алгоритм

- Дан текст T длины n и строка P длины m ;
- Левый конец образца P прикладываем к левому концу текста T ;
- Сравниваем символы слева-направо;
 - Если исчерпали P – нашли образец в тексте T ;
 - Если нашли несовпадение – сдвигаем вправо на один символ и начинаем сначала;
- Сложность по времени: $\Theta(n \cdot m)$
- Сложность по памяти: $O(1)$

Наивный алгоритм

```
naive_search(T, P):  
    n = length(T)  
    m = length(P)  
    for i = 0 to n-m; do  
        for j = 0 to m-1; do  
            if T[i+j] != P[j]; then  
                break  
            fi  
            if j == m-1; then  
                return i  
            fi  
        end for  
    end for  
    return -1
```



Упражнение #1

Реализовать наивный метод поиска образца `pattern` в тексте `text`.

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

<https://interview.cups.online/live-coding/?room=4453e993-24c0-4ade-b102-58181c1f1e87>

Z-блоки



Препроцессинг (1)

Для данной строки S и позиции $i > 1$, определим $Z_i(S)$ как длины наибольшей подстроки S , которая начинается в i и совпадает с префиксом S .

$S = \text{aabcaabxaz}$

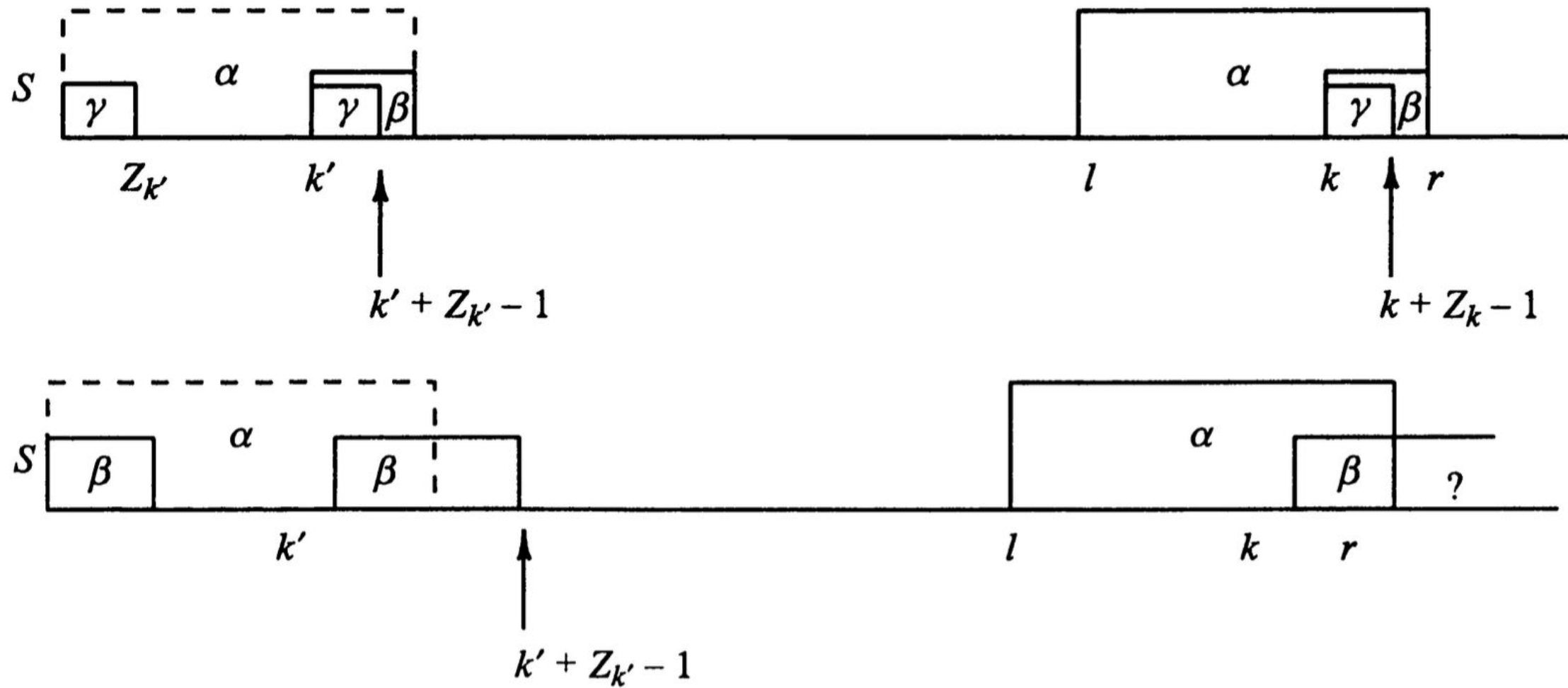
12345678901

$Z_5(S) = 3$ (aab...aabx)	aabcaabxaz
$Z_6(S) = 1$	abcaabxaz
$Z_7(S) = Z_8(S) = 0$	abcaabxaz
$Z_9(S) = 2$ (aab...aaz)	abcaabxaz

Препроцессинг (2)

- Прямое вычисление, основанное на определении, даёт время $O(|S|^2)$;
- Нужно придумать способ, который вычислял $Z_i(S)$ за $O(|S|)$;
- Для любой позиции $i > 1$, в которой $Z_i > 0$, определим Z-блок в i как интервал, начинающийся в i и кончающийся в позиции $i + Z_i - 1$.
- Для любого $i > 1$ пусть r_i — крайний правый конец Z-блоков, начинающихся не позднее позиции i . По-другому r_i можно определить как наибольшее значение $j + Z_j - 1$ по всем $1 < j \leq i$, для которых $Z_j > 0$.

Препроцессинг (3)



Преппроцессинг (4)

- Есть текст T и образец P , $m = |T|$, $n = |P|$
- Создадим строку $S = P\$T$, $\$$ - символ, отсутствующий в обеих строках;
- Посчитаем $Z_i(S)$, хранить причём будем только для P .
- Если $Z_i(s) = n$, тогда есть вхождение!
- Время работы $O(n+m) \rightarrow O(n)$

Упражнение #2

Дана строка `text` и `goal`. Определить является ли `text` циклическим сдвигом `goal`.

```
def rotate_string(text: str, goal: str) -> bool:  
    pass
```

<https://interview.cups.online/live-coding/?room=c22ab67b-4923-4206-993b-132b0ee48efe>

leetcode: <https://leetcode.com/problems/rotate-string/description/>

Упражнение #3

Реализовать алгоритм поиска подстроки, основанный на Z-блоках.

```
def z_algorithm(text: str, patter: str) -> bool:  
    pass
```

<https://interview.cups.online/live-coding/?room=d472e1a3-ff9f-4f4b-b96d-2f234478d8ef>

Бойер-Мур



Идеи

1. Просмотр справа налево,
2. Правило сдвига по плохому символу,
3. Правило сдвига по хорошему суффиксу.

12345678901234567

T: xpbctbxabpqbctbpq

P: tpabxab

1234567



Правило плохого символа

Для каждого символа алфавита x пусть $R(x)$ – позиция крайнего правого вхождения x в P . если x в P не входит, $R(x)$ считается нулём.

Правило плохого символа

Для каждого символа алфавита x пусть $R(x)$ – позиция крайнего правого вхождения x в P . если x в P не входит, $R(x)$ считается нулём.

Правило плохого символа гласит, что P следует сдвинуть вправо на $\max(1, i - R(T(k)))$ мест. Таким образом, если крайнее правое вхождение в P символа $T(k)$ занимает позицию $j < i$, то P сдвигается так, чтобы символ j в P поравнялся с символом k в T . В противном случае P сдвигается на одну позицию.

Правило плохого символа

12345678901234567

T: xpbct**t****b****x****a****b**pqxctbpq

P: tp**a****b****x****a****b**

1234567

k = 5, i = 3

R('t') = 1

$\max(1, i - R(T(k))) = \max(1, 3 - R('t')) = \max(1, 3 - 1) = 2$



Правило хорошего суффикса

123456789012345678

T: prstabstu**ba**bvqxrst

P: qc**a**b**d**a**b**

1234567890



Преппроцессинг для правила хорошего суффикса

Для каждого i пусть $L(i)$ — наибольшая позиция, меньшая n и такая, что строка $P[i..n]$ совпадает с суффиксом строки $P[1..L(i)]$.

123456789

P: cabdabdab

$L(8) = 6$

$L'(8) = 3.$

Препроцессинг для правила хорошего суффикса

Пусть $N_j(P)$ - длина наибольшего суффикса подстроки $P[1 \dots j]$, который является также суффиксом полной строки P .

Пусть $l'(i)$ обозначает длину наибольшего суффикса $P[i \dots n]$, который является префиксом P , если такой существует. Если же не существует, то $l'(i)$ равно нулю.

Алгоритм Бойера-Мура

{Стадия поиска}

$k := n$;

while $k \leq m$ do begin

$i := n$;

$h := k$;

 while $i > 0$ и $P(i) = T(h)$ do begin

$i := i - 1$

$h := h - 1$;

 end;

 if $i = 0$ then begin

 зафиксировать вхождение P в T с последней позицией k . $k := k + n - l'(2)$

 end

 else

 сдвинуть P (увеличить k) на максимальную из величин, задаваемых (расширенным) правилом плохого символа и правилом хорошего суффикса.

end;



Домашнее задание



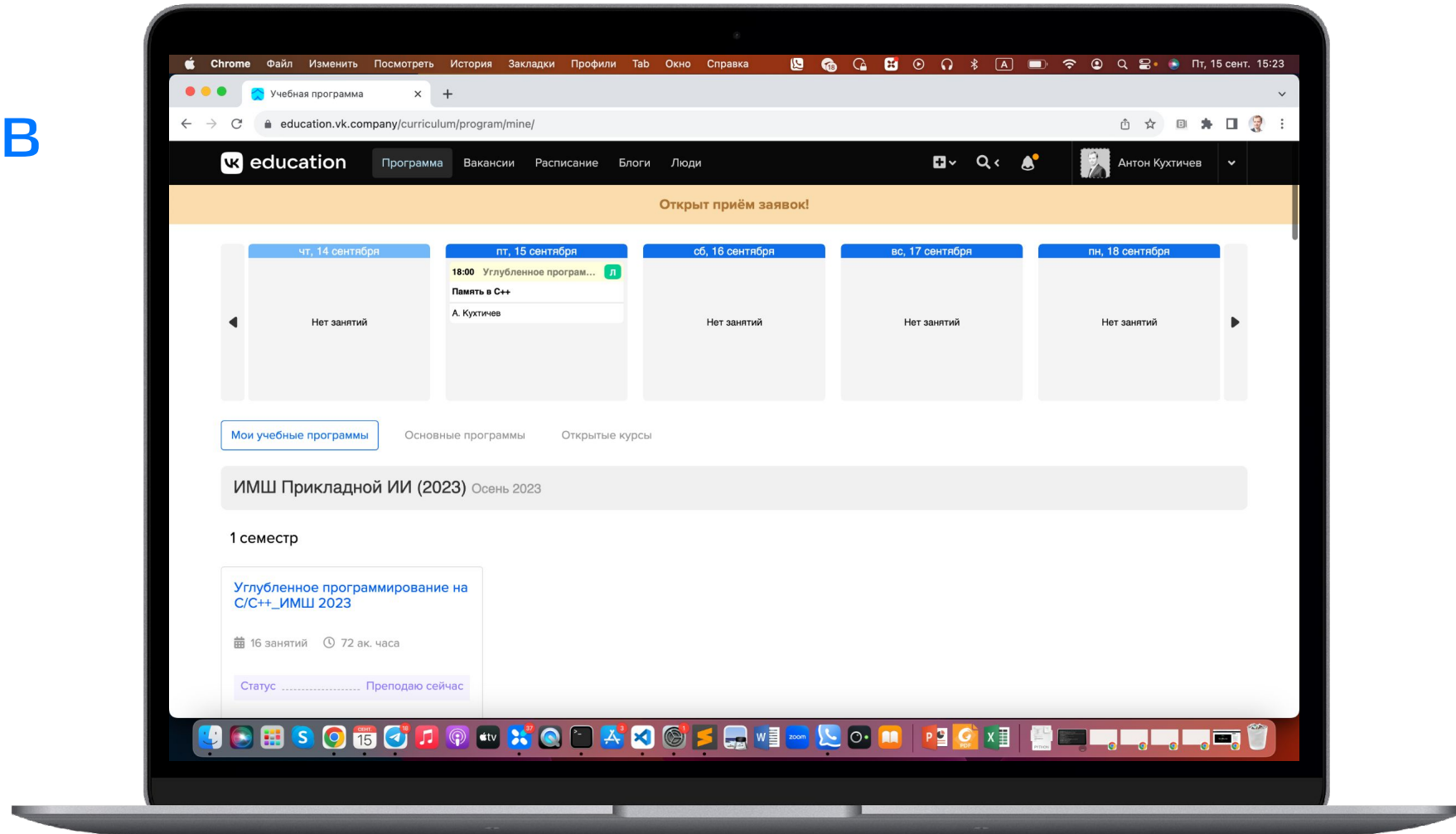
Домашнее задание

Необходимо реализовать алгоритм Бойера-Мура поиска образцов для указанного алфавита.

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

Напоминание оставить отзыв

Это правда важно





Спасибо
за внимание!