Санкт-Петербургский национальный исследовательский университет

Информационных технологий, механики и оптики

# Отчет по решению задач девятой недели
# По курсу «Алгоритмы и структуры данных»

# на Openedu

Выполнил: Сыроватский Павел Валентинович

Группа P3218

Санкт-Петербург

2019

# Наивный поиск подстроки в строке

0 возможных балла (не оценивается)

| | |
|---|---|
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Даны строки $p$ и $t$. Требуется найти все вхождения строки $p$ в строку $t$ в качестве подстроки.

*Формат входного файла*

Первая строка входного файла содержит $p$, вторая — $t$ ($1 \le |p|, |t| \le 10^4$). Строки состоят из букв латинского алфавита.

*Формат выходного файла*

В первой строке выведите число вхождений строки $p$ в строку $t$. Во второй строке выведите в возрастающем порядке номера символов строки $t$, с которых начинаются вхождения $p$. Символы нумеруются с единицы.

## Реализация на C++

```cpp
#include "edx-io.hpp"
#include <string>
using namespace std;

//Перебором каждого символа находит первое вхождение строки P в T с позиции curr_pos
int find_substr(string P, string T, long curr_pos) {
	for (int i = curr_pos; i < T.length(); i++) {
		bool ok = true;
		for (int j = 0; j < P.length(); j++)
		{
			if (T[i + j] != P[j]) {
				ok = false;
				break;
			}
		}
		if (ok) {
			curr_pos = i;
			return curr_pos;
		}
	}
	return T.length();
}

int main() {

	string P, T;
	io >> P >> T;
	int counter = 0;
	int curr_pos = 0;
	int* positions = new int[T.length()];

	do {
		//Получаем позицию вхождения
		curr_pos = find_substr(P, T, curr_pos);
		//Если метод вернул значение принадлежащее строке - записываем его
		if (curr_pos < T.length()) {
			positions[counter++] = ++curr_pos;
		}
	} while (curr_pos < T.length());

	io << counter << "\n";
	for (int i = 0; i < counter; i++) {
		io << positions[i] << " ";
	}
	return 0;
}
```

Результат выполнения первой задачи

| № теста | Результат | Время, с | Память | Размер входного файла | Размер выходного файла |
|---|---|---|---|---|---|
| Max | | 0.015 | 2383872 | 25 | 11 |
| 1 | OK | 0.000 | 2371584 | 7 | 2 |
| 2 | OK | 0.015 | 2367488 | 8 | 3 |
| 3 | OK | 0.000 | 2383872 | 5 | 1 |
| 4 | OK | 0.015 | 2371584 | 5 | 1 |
| 5 | OK | 0.000 | 2383872 | 6 | 1 |
| 6 | OK | 0.000 | 2371584 | 9 | 4 |
| 7 | OK | 0.000 | 2367488 | 23 | 10 |
| 8 | OK | 0.000 | 2371584 | 25 | 11 |
| 9 | OK | 0.000 | 2371584 | 24 | 1 |
| 10 | OK | 0.000 | 2371584 | 24 | 1 |
| 11 | OK | 0.000 | 2371584 | 14 | 10 |
| 12 | OK | 0.015 | 2371584 | 23 | 10 |
| 13 | OK | 0.015 | 2371584 | 23 | 11 |
| 14 | OK | 0.000 | 2371584 | 20 | 9 |
| 15 | OK | 0.015 | 2371584 | 23 | 11 |
| 16 | OK | 0.015 | 2371584 | 20 | 9 |
| 17 | OK | 0.000 | 2371584 | 22 | 10 |
| 18 | OK | 0.000 | 2367488 | 23 | 11 |
| 19 | OK | 0.000 | 2371584 | 22 | 10 |
| 20 | OK | 0.000 | 2371584 | 22 | 10 |
| 21 | OK | 0.015 | 2371584 | 22 | 10 |

# Наивный поиск подстроки в строке

Даны строки $p$ и $t$. Требуется найти все вхождения строки $p$ в строку $t$ в качестве подстроки.

*Формат входного файла*

Первая строка входного файла содержит $p$, вторая — $t$ ($1 \le |p|, |t| \le 104$). Строки состоят из букв латинского алфавита.

*Формат выходного файла*

В первой строке выведите число вхождений строки $p$ в строку $t$. Во второй строке выведите в возрастающем порядке номера символов строки $t$, с которых начинаются вхождения $p$. Символы нумеруются с единицы.

*Примеры*

### Реализация программы на C++

```
#include <fstream>
#include <string>
#include <map>
#include <vector>

using namespace std;
long long calculate_sum_distance(vector<long> positions) {
	long long sum = 0;
	long long temp;
	long k = positions.size() - 1;
	for (long i = positions.size() - 1; i >= 0; i--) {
		temp = (long long)k * positions[i];
		sum += temp;
		k -= 2;
	}
	for (long i = 1; i < positions.size(); i++)
	{
		sum -= i;
	}
	return sum;
}


int main() {

	ifstream input("input.txt");
	ofstream output("output.txt");

	string P = "";
	string tmp;
	while (!input.eof()) {
		input >> tmp;
		if (tmp != "") {
			P += tmp;
		}
		tmp = "";
	}
	map<char, vector<long>> letters;
	for (long i = 0; i < P.length(); i++) {
		letters[P[i]].push_back(i);
	}
	long long counter = 0;
	for (char i = 'a'; i <= 'z'; i++) {
```

```
            if (letters.count(i) && letters[i].size() > 1) {
                counter += calculate_sum_distance(letters[i]);
            }
        }

        output << counter;
        return 0;
}
```

Результат решения задачи

| № теста | Результат | Время, с | Память | Размер входного файла | Размер выходного файла |
|---|---|---|---|---|---|
| Max | | 0.031 | 2387968 | 25 | 19 |
| 1 | OK | 0.015 | 2367488 | 7 | 3 |
| 2 | OK | 0.031 | 2371584 | 8 | 3 |
| 3 | OK | 0.000 | 2371584 | 5 | 1 |
| 4 | OK | 0.015 | 2371584 | 5 | 1 |
| 5 | OK | 0.000 | 2383872 | 6 | 1 |
| 6 | OK | 0.015 | 2371584 | 6 | 1 |
| 7 | OK | 0.000 | 2371584 | 23 | 19 |
| 8 | OK | 0.015 | 2367488 | 25 | 18 |
| 9 | OK | 0.000 | 2387968 | 24 | 18 |
| 10 | OK | 0.000 | 2383872 | 24 | 19 |
| 11 | OK | 0.015 | 2371584 | 23 | 18 |
| 12 | OK | 0.015 | 2371584 | 23 | 18 |
| 13 | OK | 0.000 | 2371584 | 20 | 15 |
| 14 | OK | 0.000 | 2383872 | 23 | 18 |
| 15 | OK | 0.000 | 2371584 | 20 | 18 |
| 16 | OK | 0.000 | 2367488 | 22 | 18 |
| 17 | OK | 0.000 | 2383872 | 23 | 18 |
| 18 | OK | 0.000 | 2371584 | 22 | 17 |
| 19 | OK | 0.000 | 2367488 | 22 | 17 |
| 20 | OK | 0.000 | 2367488 | 22 | 18 |