Санкт-Петербургский Научный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Задачи девятой недели

по курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: студент группы P3218

Артамонов Александр Владимирович

Санкт-Петербург, 2019г

Задача 1. 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 ($1 \le |p|,|t| \le 10^4$). 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.

Примеры

input.txt	output.txt
aba abaCaba	2 1 5

Код программы (C++)

```
//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;
}
```

Бенчмарк (задача 1)

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.125	2437120	20003	48890
1	OK	0.000	2220032	14	7

Задача 2. Карта

В далеком 1744 году во время долгого плавания в руки капитана Александра Смоллетта попала древняя карта с указанием местонахождения сокровищ. Однако расшифровать ее содержание было не так уж и просто.

Команда Александра Смоллетта догадалась, что сокровища находятся на x шагов восточнее красного креста, однако определить значение числа она не смогла. По возвращению на материк Александр Смоллетт решил обратиться за помощью в расшифровке послания к знакомому мудрецу. Мудрец поведал, что данное послание таит за собой некоторое число. Для вычисления этого числа необходимо было удалить все пробелы между словами, а потом посчитать количество способов вычеркнуть все буквы кроме трех так, чтобы полученное слово из трех букв одинаково читалось слева направо и справа налево.

Александр Смоллетт догадывался, что число, зашифрованное в послании, и есть число x. Однако, вычислить это число у него не получилось.

После смерти капитана карта была безнадежно утеряна до тех пор, пока не оказалась в ваших руках. Вы уже знаете все секреты, осталось только вычислить число x.

Формат входного файла

В единственной строке входного файла дано послание, написанное на карте. Длина послания не превышает $3 \cdot 10^5$. Гарантируется, что послание может содержать только строчные буквы английского алфавита и пробелы. Также гарантируется, что послание не пусто. Послание не может начинаться с пробела или заканчиваться им.

Формат выходного файла

Выведите одно число x — число способов вычеркнуть из послания все буквы кроме трех так, чтобы оставшееся слово одинаково читалось слева направо и справа налево.

Примеры

input.txt	output.txt
treasure	8
you will never find the treasure	146

Код программы (C++)

```
//Находит сумму дистанций - разностей позиций каждого с каждым
//Формула: сумм(((n-1) - 2(i-1)) * pos[i - 1])
//i изменяется так, чтобы pos[i] > pos[i+1]
long long calculate_sum_distance(vector<long> positions) {
	long long sum = 0;
	long long temp;
	//Коэффициент слагаемого (n-1) - 2(i-1)
	long k = positions.size() - 1;
	for (long i = positions.size() - 1; i >= 0; i--) {
		temp = (long long)k * positions[i];
		sum += temp;
		k -= 2;
	}
	//Точки не включены, поэтому необходимо из каждой разности вычесть единицу
	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;
	//Слепливаем строку в одну без пробелов
	while (!input.eof()) {
		input >> tmp;
		if (tmp != "") {
			P += tmp;
		}
		tmp = "";
	}
	//Ассоциативный массив списков, отсортированный по алфавиту
	map<char, vector<long>> letters;
	//Позицию каждой буквы в строке добавляем в соответствующий список
	for (long i = 0; i < P.length(); i++) {
		letters[P[i]].push_back(i);
	}
	long long counter = 0;
	//Для каждого списка с количесвом элементов > 1 находим сумму разностей его
элементов
	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;
}
```

Бенчмарк (задача 2)

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	5971968	300002	16
1	OK	0.031	2363392	10	1