Домашнее задание 10. Простые строковые алгоритмы
Автор: *Головко Денис*, Б05–225

Задача 2.

Решение.

Докажем утверждение по индукции. Для $n = 1$ утверждение верно. Предположим, что для n утверждение доказано. Рассмотрим строку $s = a_1 \dots a_n$. Добавим к ней символ x, получим $sx = a_1 \dots a_n x$. Покажем, что подпалиндромов увеличилось не более чем на один. Рассмотрим два случая:

1. Пусть $x \notin \{a_1, \dots, a_n\}$. Тогда появился один новый подпалиндром "x".
2. Пусть $x \in \{a_1, \dots, a_n\}$ и появилось хотя бы два новых подпалиндрома:

   $$\exists k, l : 1 \leqslant k \leqslant l \leqslant n \, (a_k = a_l = x, \text{ "}a_k a_{k+1} \dots x\text{" и "}a_l a_{l+1} \dots x\text{" – новые подпалиндромы})$$

   Тогда $\exists j : k < j \leqslant n : a_k \dots a_j = a_l \dots x$, то есть подпалиндром "$a_l \dots x$" уже существовал. Значит, новых подпалиндромов появилось не более одного.

Переход индукции доказан.

Задача 3.

Решение.

Сначала каждому массиву однозначно сопоставим разностный массив. Например, массиву $a = [5, 6, 8, 9, 1, 2, 4]$ будет сопоставлен $\text{diff}_\text{a} = [-1, -2, -1, 8, -1, -2]$, а массиву $b = [1, 2, 4]$ сопоставим $\text{diff}_\text{b} = [-1, -2]$. Это занимает $O(|a| + |b|)$ времени. Далее, осталось найти число вхождений массива diff_b в массив diff_a.

Построим префикс-функцию для diff_b, это позволит избежать лишних сравнений при попытке сопоставить diff_b с подмассивами diff_a. Далее пробежимся по массиву diff_a, сохраняя j – позицию в diff_b. При возникновении несовпадения префикс-функция покажет, в каком месте diff_b следует продолжить сравнение. Приведу псевдокод решения (обозначим $a = \text{diff}_\text{a}$, $b = \text{diff}_\text{b}$):

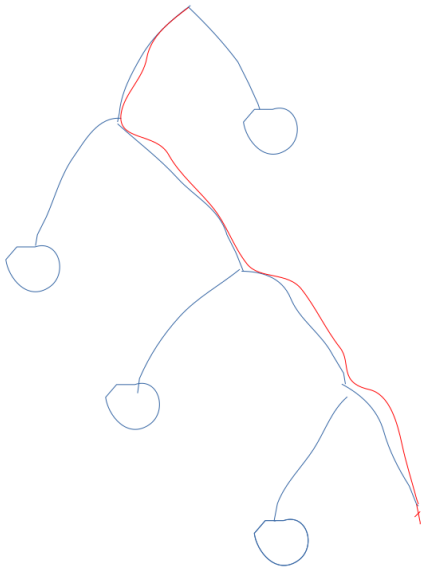
```
1.    int count = 0;
2.    std::vector<int> pi = compute_prefix_function(b);
3.    int j = 0;
4.    for (int i = 0; i < length(a); i++) {
5.        while (j > 0 && (j == length(b) || a[i] != b[j])) j = pi[j-1];
6.        if (a[i] == b[j]) ++j;
7.        if (j == len(b)) {
8.            ++count;
9.            j = pi[j-1];
10.       }
11.   }
12.   return count;
```

Асимптотика. $O(|a| + |b|)$

Задача 4.

Решение.

Пусть h – высота сжатого бора (максимальная длина ветки). Покажем, что $h = O\left(\sqrt{\sum |s_i|}\right)$. Рассмотрим самую длинную ветку в сжатом боре, ее высота h. Рассмотрим лист. Так как бор сжатый, исходящая степень вершины перед листом хотя бы 2. Аналогично, поднимаемся наверх. Тогда сумма высот всех веток будет хотя бы $1 + 2 + \ldots + h = \frac{h(h+1)}{2}$. (На рисунке красная ветка – самая длинная высоты остальных, начиная от корня, оцениваются хотя бы 1, хотя бы 2, и тд.)



Тогда, имеем оценку: $\Theta(h^2) \leqslant \sum |s_i|$. Отсюда следует, что $h = O\left(\sqrt{\sum |s_i|}\right)$.

Оценим количество вершин в графе. Рассмотрим произвольное дерево на n вершинах. В нем $n - 1$ ребро. Сумма степеней вершин в таком графе $2n - 2$ (так как каждое ребро добавляет 2). Получим, что средняя степень вершины в дереве $\frac{2n-2}{n} = 2 - \frac{2}{n} < 2$. Теперь рассмотрим сжатый бор. В нем есть либо вершины степени 1, либо вершины степени хотя бы 3. Вершин степени 1 всего $O(n)$, где n – количество строк. Из доказанного ранее факта, так как средняя степень вершины в дереве меньше двух, вершин степени хотя бы 3 так же $O(n)$. Значит, количество вершин в графе можно оценить как $O(n)$.

Задача 5.

Решение.

Вначале применим алгоритм Ахо-Корасика к строкам первой таблицы. Используя этот алгоритм, найдем все вхождения строк из первой таблицы в строки второй таблицы. После этого в соответствующих местах второй таблицы будем сохранять номера терминальных вершин бора. Последний этап — искать в столбцах второй таблицы последовательности номеров $t_0 t_1 \ldots t_{n-1}$, где каждое t_i — это номер терминальной вершины бора для i-ой строки из первой таблицы. Поиск можем выполнить с применением Z-функции. В итоге, время выполнения будет линейно зависеть от площадей данных таблиц.