Алгоритмы и структуры данных. Домашнее задание №9

Выполнил студент Наседкин Дмитрий Сергеевич (группа 242)

Письменная часть

№ 1

Идея: разобьем строку-паттерн p по символам * на непустые отрезки (если идут 2 подряд *, то это не имеет смысла, их можно заменить на одну), назовем их $t_1, t_2, \ldots, t_n$, тогда проверка вхождения p в s как подстроки выглядит следующим образом:

- Выбираются $1 \leq pos_1 < pos_2 < \cdots < pos_n \leq |s|$, где $pos_{i+1} - pos_i >= |t_i|$ (чтобы последняя тоже поместилась, будем считать, что $pos_{n+1} = |s| + 1$).
- Для любого $i \in [1, n]$ верно, что $s_{\text{pos}_i} = t_{i_1}, s_{\text{pos}_i+1} = t_{i_2} \ldots, s_{\text{pos}_i+|t_i|} = t_{i_{|t_i|}}$, другими словами - вся строка t_i совпадает с подстрокой s начиная с pos_i.

Заметим, что можно действовать жадно: выбирать первое подходящее pos_1, отступить на $|t_1|$ вправо и начинать искать первое подходящее pos_2 и т.д. Если нашли все t_i, то вхождение выполняется, иначе нет.

Докажем оптимальность такого подхода: Пусть существует какое-то вхождение p в s. Разобьем точно также p по звездочкам и найдем $pos_1, pos_2, \ldots, pos_n$ в s, которым они соответствуют в s. Заметим, что ничего не мешает сдвинуть pos_1 влево, если там есть вхождение, тогда выберем новое pos_1 как можно ближе к началу. Точно также можно двигать и pos_2 (но уже не до начала строки, а до $pos'_1 + |t_1|$) и т.д. Итого получили бы строку, которую нашел бы наш алгоритм.

Реализация:

Разобьем строку-паттерн p по символам * на непустые отрезки, соберем это в вектор строк t. Очевидно что это делается за $O(|p|)$.

Теперь будем искать pos_i для каждого i от 1 до n по строке s с помощью КМП. Изначально ищем t_1 с позиции 1 в s. После того как нашли очередное вхождение t_i в позиции k, то t_{i+1} начинаем искать с позиции $k + 1$. Если мы дошли до конца строки s найдя не все мини-строки, то говорим, что вхождения нет, иначе да (и ничего не мешает также сказать позиции вхождения всех мини-строк). Время работы такого КМП - мы пройдемся один раз по всем мини-строкам (это суммарно $|p|$) + по всей s один раз(это суммарно $|s|$) + n символов # (это не более $|p|$).

Получили требуемое $O(|p| + |s|)$.

№ 2

(a)

Пусть параметр k — целое. Возьмём все строки вида

$$s_{i,\ell} = i; i + 1; i + 2; \ldots; i + \ell - 1,$$

для $1 \leq i \leq k$ и $1 \leq \ell \leq k$. Можно думать, что каждое «чисел» i — это отдельный символ из большого алфавита, тогда все такие строки различныю. См. ниже для большего понимания

start \len	1	2	3	4	5	6	7
1	1	12	123	1234	12345	123456	1234567
2	2	23	234	2345	23456	234567	2345678
3	3	34	345	3456	34567	345678	3456789
4	4	45	456	4567	45678	456789	45678910
5	5	56	567	5678	56789	5678910	567891011
6	6	67	678	6789	678910	67891011	6789101112
7	7	78	789	78910	7891011	789101112	78910111213

1. **Суммарная длина (L).**
   Длина строки $s_{i,\ell}$ равна ℓ. Значит

$$L = \sum_{i=1}^{k} \sum_{\ell=1}^{k} \ell = k \cdot \frac{k(k+1)}{2} = \frac{k^2(k+1)}{2} \sim k^3.$$

2. **Количество пар $(i_1, \ell_1), (i_2, \ell_2)$ таких, что одна — подстрока другой.**
   Фиксируем хозяина $s_{b,m}$ (начало b, длина m). Какие строки содержатся в нём? Это просто все подстроки строки-хозяина. Их кол-во:

$$\#\text{строки в } s_{b,m} = \sum_{t=1}^{m} t = \frac{m(m+1)}{2}.$$

Теперь суммируем по всем хозяевам (b=1..k,; m=1..k):

$$P = \sum_{b=1}^{k} \sum_{m=1}^{k} \frac{m(m+1)}{2} = k \cdot \frac{1}{2}\Big(\sum_{m=1}^{k} m^2 + \sum_{m=1}^{k} m \Big) \sim k^4$$

Тогда выбрав $k \sim L^{1/3}$ получим число пар, равное $c \cdot L^{\frac{4}{3}}$. Получили требуемое.

Письменная и устная часть

(b)
Пусть

$$P = \#(i,j) : s_i \text{ является подстрокой } s_j.$$

Построим оценку через параметр K.
Пусть K — произвольное целое ≥ 1.
Разделим все строки s_i на короткие (длина $\leq K$) и длинные (длина $> K$). Оценим вклад от каждой группы в P.

1. **Вклад коротких строк**.
   Зафиксируем строку-хозяина s_j длины m. Для каждой длины $t \leq K$ число различных подстрок длины t не превышает m. Значит, суммарно по всем $t \leq K$ не более Km. Тогда по всем j получаем вклад от коротких не больше

$$\sum_{j} K|s_j| = KL.$$

2. **Вклад длинных строк**.
   Пусть s_i — фиксированная строка длины $m > K$. Найдм кол-во строк, что могут содержать s_i. Суммрная длина всех вхождений (уникальных - мы не считаем если одна строка содержит вторую 2 раза) s_i в s_j равна (число хозяев)$*m$ и меньше либо равна длины всех строк,

т.е. $\leq L$. Отсюда число хозяев, содержащих s_i, не больше $\frac{L}{m} \leq \frac{L}{K}$. И так как длинных строк не больше $\frac{L}{K}$. То получаем вклад $\frac{L^2}{K^2}$

$$P \leq KL + \frac{L^2}{K^2}.$$

Выбирем K так, чтобы правая часть была как можно меньше. Получаем $K \sim L^{1/3}$.

$$P \leq O(LL^{1/3}) + O(\frac{L^2}{L^{2/3}}) = O(L^{4/3}).$$

№ 5

Построим суффиксный автомат по строке s за время $O(|s|)$. Напомню, что каждое состояние суф. автомата v хранит 2 величины:

- $len[v]$ — длина максимального образа (longest string) в классе состояния v;
- $link[v]$ — суффиксная ссылка;

Тогда минимальная длина образа в этом состоянии очевидно равна $len[link[v]] + 1$.

Все разные подстроки строки s ставятся в биекцию с парами (`state v, length L`), где $L \in [len[link[v]] + 1, \ldots, len[v]]$. То есть каждое состояние соответствует набору подряд идущих длин подстрок; для каждой такой длины это ровно одна различная подстрока.

(Все вышесказанное так или иначе было на лекции или семинарах).

А значит, чтобы получить для каждого k число разных подстрок длины k, достаточно для каждого состояния $v \neq 0$ прибавить $+1$ к всем длинам L в интервале $[len[link[v]] + 1, len[v]]$. Кол-во состояний $O(n)$ (если быть точным не более $2n$, тоже с лекции), то есть нужно сделать не более $2n$ прибавлений $+1$ на отрезке. Это делается в офлайне с помощью разностного массива: $diff[l] += 1; diff[r + 1] -= 1$. После подсчета префикс-сумм на разностном массиве получим ответ для всех k. Очевидно что это работает за $O(|s|)$.