Жадная гипотеза в задаче о надстроке. Задача 1

12 марта 2023 г.

$\forall$ s, X: s - строка, X - символ, введем функции:

1. $len(s)$ - возвращает длина строки s
2. $len_1(s, X)$ - возвращает количество символов X в строке s
3. $add(s, X)$ - возвращает строку t, где $t = s_1 + X + s_2 + X + ... + S_n + X$ (другими словами, после каждого символа строки s добавляем символ X)
4. $remove(s, X)$ - возвращает строку s без X в ней (удаляем все символы X)

Пусть $S = s_1, s_2, ..., s_n$ - множество строк для которого жадный алгоритм (далее алгоритм 1) для задачи о надстроке строит в α раз более длинное решение, чем оптимальное.

Тогда рассмотрим множество строк $T = t_1, t_2, ..., t_n$, в котором $t_i = add(s_i, X)$ (где X не содержится ни в одной из строк множества S, X - один и тот же для каждой строки) $\forall i \in \mathbf{N}$: i <= n.

Докажем, что для множества строк T и символа X жадный алгоритм для второй задачи (далее алгоритм 2) построит по крайней мере в α раз более длинное решение, чем оптимальное.

Теорема 1: Для множеств $S = s_1, s_2, ..., s_n$ и $T = t_1, t_2, ..., t_n$, в котором $t_i = add(s_i, X) \forall i \in \mathbf{N} : i <= n$, (причем X не содержится ни в одной из строк S) Алгоритм 1 (выполняется на S) выберет строки для слияния k_1, k_2, а Алгоритм 2 (выполняется на T, с символом X) выберет строки для слияния l_1, l_2, такие что $k_1 = remove(l_1, X)$, $k_2 = remove(l_2, X)$ (после выполнения шага алгоритма 1 и алгоритма 2, получатся одинаковые множества строк, если из каждой строки из T удалить X)

Доказательство:

1. Алгоритм 1 выберет строки с наибольшей длиной пересечения, а Алгоритм 2 выберет строки с наибольшим количеством X в пересечении. Но так как длина пересечения строк из S равна количеству X в соответсвующих строках из T, то алгоритмы выберут одинаковые строки (при удалении X-ов). Действительно, если $s_i = A + B, s_j = B + C$ (A, B, C - строки, B - максимальна по длине), то длина их пересечения равна $len(B)$, а для t_i и t_j: $t_i = A_1 + X + A_2 + X + ... + A_m + X + B_1 + X + B_2 + X + ... + B_k + X$, $t_j = B_1+X+B_2+X+...+B_k+X+C_1+X+C_2+X+...+C_l+X$ их пересечение как видно из разложения равно $B_1+X+B_2+X+...+B_k+X$, количество X равно len(B). Получили, что алгоритмы выбирают одинаковые строки для S и T.

2. При этом, при объединении строк получим (без ограничения общности считаем, что строки с индексами i и j выбраны алгоритмами для слияния): для алгоритма 1: s* = A + B + C, а для алгоритма 2: t* = $A_1 + X + A_2 + X + ... + A_m + X + B_1 + X + B_2 + X + ... + B_k + X + C_1 + X + C_2 + X + ... + C_l + X$, s* = remove(t*, X).

3. Также заметим, что новые строки также удовлетворяют условию, t* = add(s*, X). Т.е. после одного шага алгоритма, Теорема 1 справедлива для новых полученных множеств.

Что и требовалось доказать

Теорема 2: Пусть для множеств $S = s_1, s_2, ..., s_n$ и $T = t_1, t_2, ..., t_n$, в котором $t_i = add(s_i, X) \forall i \in \mathbf{N} : i <= n$ (причем X не содержится ни в одной из строк S), s_{opt} - оптимальная надстрока для S, а t_{opt} - оптимальная надстрока для T. Тогда $len_1(t_{opt}) \leq len(s_{opt})$.

Доказательство:

1. Рассмотрим $s_1 = add(s_{opt}, X)$, тогда $add(s_i, X)$ - подстрока S $\forall i \in \mathbf{N} : i <= n$ (так как каждый второй символ в s_{opt} - X, и s_{opt} содержал в качестве подстрок все строки из S, но так как и в $add(s_i, X)$ каждый второй символ - X, то утверждение верно). Значит s_1 - надстрока для T (так как $add(s_i, X) = t_i$ по условию).

2. $len_1(s_1) = len(s_{opt})$ (X после каждого символа, значит количество X равно количеству символов в s_{opt}). Получили, что существует надстрока для T, причем $len_1(s_1) = len(s_{opt})$, значит $len_1(t_{opt}) \leq len_1(s_1) = len(s_{opt})$ (так как в оптимальной строке символов X точно не больше, чем в s_1)

Что и требовалось доказать

Нетрудно заметить, что после каждого шага обработки алгоритмами 1 (на множестве S) и 2 (на множестве T и символе X) мы будем получать одни и те же множества строк (при удалении X-ов из строк T), по Теореме 1.

Таким образом в конце получатся строки s_{end} и t_{end}: $s_{end} \in S$ и $t_{end} \in T$. Причем $t_{end} = add(s_{end}, X)$ (по Теореме 1, пункт 3), значит $len_1(t_{end}) = len(s_{end})$ (X стоят после каждого символа в s_{end}, значит их количество равно количеству символов в s_{end}).

Пусть строка s_{opt} - оптимальная для множества S, а строка t_{opt} - оптимальная для множества T и символа X. По условию задачи $len(s_{end}) = \alpha \cdot len(s_{opt})$, но так как $len_1(t_{opt}) \leq len(s_{opt})$ (по Теореме 2) и $len_1(t_{end}) = len(s_{end})$, то верно, что $len_1(t_{end}) \geq len_1(t_{opt}) * \alpha$, получилил что для множества строк T и символа X алгоритм 2 построит в по крайней мере α раз длинное решение, чем оптимальное (все выводы заключаются на основе теорем и том факте, что всегда для строк S, T верно: $t_i = add(s_i, X) \forall i \in \mathbf{N} : i <= S$ по изначальному выбору S и T и Теореме 1, пункт 3)

Что и требовалось доказать