Второй курс, осенний семестр 2021/22
Практика по алгоритмам #9

Ахо-Корасик, суфдерево

21 ноября

Собрано 25 ноября 2022 г. в 02:08

Содержание

1. Ахо-Корасик, суфдерево **1**

2. Разбор задач практики **3**

3. Домашнее задание **6**

3.1. Обязательная часть 6

3.2. Дополнительная часть 6

Ахо-Корасик, суфдерево

1. **Сравнение множеств (задача из дз)**

   Придумайте хеш-функцию, которая позволяет за $\mathcal{O}(1)$ делать 3 операции с множествами целых положительных 32-битных чисел.

   a) Добавить элемент в множество
   b) Удалить элемент из множества
   c) Проверить два множества на равенство.
   d) `(*)` `+=` ко всем элементам множества.

2. **ST → SA**

   По суффиксному дереву постройте суффиксный массив с `LCP`.

3. **SA → ST**

   По суффиксному массиву с LCP постройте суффиксное дерево.

4. **Словари offline**

   Даны словарь (конечное множество слов) и текст. Пример: $\{abcd, bc, a, da\}$ и текст $abcda$.

   a) Для каждого слова из словаря определить, входит ли оно как подстрока в текст.
   b) Для каждого слова из словаря найти число вхождений в текст.

5. **Словари online**

   Даны словарь (конечное множество слов) и текст. Обновлять ответ `online` при добавлении символа в конец текста.

   a) Пересчитать суммарное число вхождений слов из словаря в текст за $\mathcal{O}(1)$.
   b) Пересчитать множество всех вхождений слов из словаря в текст за $\mathcal{O}(1 + |\Delta A|)$, где ΔA – приращение ответа после добавления очередного символа.
   c) `(*)` за $\mathcal{O}(\log n)$ для все слов обновить число вхождений.

6. **Разбиение на словарные слова**

   Дан словарь слов суммарной длины L и текст T. Слова длины $\leqslant l$.

   a) Представить T в виде конкатенации минимального числа словарных слов за $\mathcal{O}(L + l|T|)$. Слова можно использовать более одного раза.
   b) `(*)` За $\mathcal{O}(L + \sqrt{L}|T|)$.
   c) Представить T в виде конкатенации min числа **подстрок** словарных слов. $\mathcal{O}(L + |T|)$

7. **Задачи про суффиксное дерево**

   a) Найти количество различных подстрок.
   b) Найти самый длинный рефрен – подстроку s: $\text{count}(s) \cdot |s| \to \max$.
   c) Самая длинная подстрока, входящая в s дважды, причём вхождения не пересекаются.
   d) Общая подстрока двух строк за $\mathcal{O}(|s| + |t|)$.
   e) Общая подстрока $k \leqslant 64$ строк за суммарную длину всех строк.
   f) `(*)` Общая подстрока $\forall k$ строк за суммарную длину всех строк.

8. **Амортизация в Укконене**

   Приведите пример, когда Укконен

   a) при добавлении одного символа потратит $\Omega(n)$ времени,
   b) (*) при переходе по суффсылке спустится вниз $\Omega(\sqrt{n})$ раз.

9. **Бажный Укконен**

   При подсчёте суффиксных ссылок в алгоритме Укконена маленький Петя делает спуск не по рёбрам (пройти всё ребро за шаг), а по символам (пройти один символ за шаг). Приведите пример строки, на которой полученный алгоритм будет работать $\omega(n)$.

10. **∞, избегаемость шаблонов («вирусы»)**

    Дан словарь слов суммарной длины L. За время $\mathcal{O}(L)$ определите, существует ли бесконечная строка, не содержащая ни одно словарное слово как подстроку.

11. **Один бор хорошо, а два – лучше!**

    Даны два бора A и B. Найдите для каждой вершины $u \in A$ самую глубокую вершину B, путь до которой равен суффиксу path: $root_A \rightsquigarrow u$. $\mathcal{O}(|A| + |B|)$. Размер алфавита $\mathcal{O}(1)$.

    Как увидеть в этой задаче обобщение Ахо-Корасик? Что есть A и B в Ахо-Корасик?

12. **(*) Быстрый Ахо-Корасик**

    Дан бор A над алфавитом Σ произвольного размера, постройте суффиксные ссылки за $\mathcal{O}(|A| \cdot poly(\log))$. На самом деле за такое время можно построить полный автомат.

    Мы уже умеем строить суффиксные ссылки несколькими способами.
    Полный автомат мы строим за $\mathcal{O}(|A| \cdot |\Sigma|)$. `bfs +` откаты, как в префикс функции, работают за $\mathcal{O}(|s_i|)$, что может быть сильно больше $|A|$.

13. **(*) Окно и стек**

    a) Найти «количество различных подстрок» с помощью суффиксного массива.
    b) Найти максимальный рефрен с помощью суффиксного массива.

Разбор задач практики

1. **Сравнение множеств (задача из дз)**

   Мы можем поддерживать для множества A величину $\sum_{a\in A} f(a) \bmod m$.

   Осталось придумать f: легко посчитать и сложно подделать. $f(a) = P^a \bmod m$.
   $\forall$`a` `a+=`Δ`a` $\Leftrightarrow$ $f(a)$`*=`$P^{\Delta \mathtt{a}}$ $\bmod m \Rightarrow$ все три операции `add`, `del`, `+=` за $\mathcal{O}(\log a)$.
   Вероятность ошибки: P случайное и должно быть корнем многочлена $\deg = \max a \Rightarrow$
   $\Pr \leqslant \frac{\max a}{m}$. $\max a \leqslant 10^9, m \leqslant 10^{18} \Rightarrow$ `OK`. Но для $A_1 = \{0\}$, $A_2 = \{m-1\}$ хеши совпадут $\forall P$.

   Мемное решение. Пусть $f(x) = \mathtt{mem}[x]$, где `mem` — хеш-таблица, куда при первом обращении кладётся псевдорандом. $\Pr[\text{ошибки}] = \frac{1}{m}$, время `add` `del` $\mathcal{O}(1)$, но много лишней памяти.

   Третий вариант. $A = \prod_{a\in A}(z-a) \bmod {}_m$, где z фиксированное случайное.
   Множества A и B имеют коллизию $\Leftrightarrow$ z — корень многочлена степени $\leqslant \max |A|, |B| \Rightarrow$
   $\Pr[\text{ошибки}] \leqslant \frac{|A|}{m}$. `add` за $\mathcal{O}(1)$, `del` за $\mathcal{O}(\log m)$ т.к. нужно обращать по модулю.

   Итого. Первое — единственное решение с `+=`. Хеш-таблица даёт самое быстрое решение, но ест много памяти. Третье решение лучше первого по ошибке и времени `add`.

2. **ST $\rightarrow$ SA**

   dfs по дереву, помним позицию самой высокой вершины с момента, как последний раз были в листе, эта высота и есть LCP.

3. **SA $\rightarrow$ ST**

   Пусть мы уже построили дерево для первых i суффиксов SA, стоим в листе. Поднимаемся от листа до уровня $lcp[i]$, создаём развилку, новый лист.

4. **Словари offline**

   Строим Ахо-Корасик для словаря.

   a) Проходим по тексту, помечаем все посещенные вершины. В конце проходим снизу вверх по бору и делаем `visited[suf[v]] |= visted[v]`.
   b) Проходим по тексту, считаем число посещений каждой вершины. В конце проходим снизу вверх по бору и делаем `count[suf[v]] += count[v]`.

5. **Словари online**

   Строим Ахо-Корасик для словаря.

   a) Посчитаем динамику: количество терминальных вершин на суффиксном пути.
   b) Посчитаем супер-суффиксные ссылки: ближайшая терминальная вершина на суффиксном пути.
   c) Link-Cut/HLD и `+=` на пути в дереве суффссылок.

6. **Разбиение на словарные слова**

   a) Строим бор из словаря. `f[i]` – минимальная стоимость выписать префикс длины `i`.
   Динамика вперёд: спускаемся по бору суффиксом `s[i:]`, если очередная вершина бора конечная, то `relax(f[i + dep], f[i] + 1)`.
   Максимальная глубина бора $l \Rightarrow$ время $\mathcal{O}(L + l|T|)$.

b) Строим суффдерево для $s_1\#s_2\#\dots\#s_n$ за $\mathcal{O}(L)$.

Жадно разбиваем текст за $\mathcal{O}(|T|)$: пока текст не пуст, отрезаем самый длинный префикс текста, по которому можем спуститься в боре.

Способ #2: Ахо-Корасиком. Он находит самую глубокую вершину бора, в которую можно прийти суффиксом текста. Считаем динамику, держим очередь с минимумом для окна, покрытого путем до текущей вершины бора.

7. **Задачи про суффиксное дерево**

В суффиксном дереве s $\forall x$ подстрока s заканчивается в вершине u или посреди ребра $v \to u$. В поддереве u заканчиваются все суффиксы начинающиеся в x (все вхождения x).

a) *Число подстрок.* В боре это число вершин. В сжатом боре это сумма длин ребер.
b) *Рефрен.* $\mathrm{count}(s)$ – число суффиксов, кончающихся в поддереве, если спуститься по s. Заметим, что достаточно смотреть на строки, кончающиеся в вершинах.
c) *Дважды входящая.* Ищем v: $\min(R[v]-L[v], \mathrm{depth}[v]) \to \max$. Где $L[v]$ – самый левый, $R[v]$ – самый правый. Ответ может быть посреди ребра.
d) *Общая подстрока.* Суфдерево для `s#t#`. Самая глубокая вершина, в поддереве которой есть и суффикс s, и суффикс t. Чтобы это определить, проверяем $L[v] < |s|, R[v] > |s|$. А можно хранить два бита — «есть ли суффикс такого типа в поддереве».
e) $k \leqslant 64$ *строк.* Для каждой вершины считаем битмаску номеров строк, суффиксы которых есть в ее поддереве. Ищем самую глубокую, у поддерева которой маска $= k$ единиц.
f) $\forall k$. За $\mathcal{O}(n)$ посчитаем для каждой вершины количество различных чисел в поддереве. Было в прошлом семестре через LCA и суммирование снизу вверх по дереву.

8. **Амортизация в Укконене**

a) $aaa\dots ab$. Добавление последнего символа создаст n новых листьев.
b) $axbbbbbbbbbbbbby$ xbc $xbbc$ $xbbbc$ $xbbbbc$ $xbbbbbc$ $\dots$ $axbbbbbbbbbbbbbz$

9. **Бажный Укконен**

$aaa\dots a$. Добавляем b. Для создания листа суффикса i нужно спуститься из корня на i.

10. **∞, избегаемость шаблонов («вирусы»)**

Строим автомат Ахо-Корасик для словаря. Запретим (выкинем из автомата) вершины, на пути из суффссылок от которых есть запрещённые слова. Проверим, есть ли в графе цикл, достижимые из стартовой вершины. Бесконечная строка $\exists$ iff $\exists$ такой цикл.

11. **Один бор хорошо, а два – лучше!**

Замкнем бор B до полного автомата за $\mathcal{O}(|B|)$. Для каждой вершины $u \in A$ ответ пересчитываем через предка: `ans[a[v][c]] = b[ans[v]][c]`.

12. **(*) Быстрый Ахо-Корасик**

Храним в каждой вершине v персистентный массив `next[v]`: `next[v] = next[suf[v]]` кроме тех рёбер, которые торчат из вершины v вниз. `suf[v] = next[suf[p[v]]][pchar[v]]`.

13. **(*) Окно и стек**

a) Ответ $= \sum_i |suf_i| - \sum_i LCP_i = \frac{1}{2}n(n+1) - sum(LCP)$

b) Взяли `LCP` соседних, получили задачу: найти отрезок такой, что (длина $\times$ min $\to$ max). Умеем её решать стеком за $\mathcal{O}(n)$ из первого семестра: перебираем $\min LCP$, наш отрезок — взять от него ближайший меньший LCP слева-справа.

Домашнее задание

3.1. Обязательная часть

1. **(2) Первое и последнее вхождение**

   Даны словарь и текст. Найдите для каждого словарного слова первое и последнее его вхождения в текст в качестве подстроки.

2. **(3) Навигация в боре**

   Даны бор A и строка s. Найдите вершину бора v, от которой строку s можно отложить вниз по бору. Размер алфавита $\mathcal{O}(1)$. Время $\mathcal{O}(|A| + |s|)$.

3. **(3) Количество строк**

   Дан словарь и число n.
   Посчитайте количество строк длины n над алфавитом $\{a, b\}$, которые не содержат ни одного словарного слова, как подстроку. Время $\mathcal{O}(nL)$, где L – суммарная длина слов в словаре.
   (+1) допбалл за $\mathcal{O}(L)$ памяти.

4. **(3) Уникальные суффиксы**

   Два запроса:

   a) `addLetter(c)` – дописать в конец строки символ c.
   b) `isUnique(len)` – является ли суффикс длины len уникальной подстрокой.

5. **(3) k-я общая подстрока**

   Найти k-ю лексикографически общую подстроку s и t за $\mathcal{O}(|s| + |t|)$. Размер алфавита $\mathcal{O}(1)$.
   Решите суффиксным деревом.
   (+1) за произвольный размер алфавита.

3.2. Дополнительная часть

1. **(2) Поиск по отрезку словаря**

   Дан словарь $s_1, s_2, \ldots, s_n$.
   Отвечать в offline ((+1) online) на запросы `get(t,l,r)` — сколько строк из $\{s_l, \ldots, s_r\}$ входят в текст t как подстроки. Время работы – линия от размера входа на полилог.

2. **(3) Динамический словарь**

   В словаре теперь добавляются «`add(s)`» и удаляются «`del(s)`» слова.
   Необходимо в `online` научиться отвечать на запрос `get(t)` вида «входит ли в текст t хоть одно словарное слово». Время работы `add(s)` и `del(s)`: $\mathcal{O}(|s| \log L)$, время работы `get(t)`: $\mathcal{O}(|t| \log L)$ (L — суммарная длина всего).

3. **(3) Три вхождения**

   Дана строка s. Найти число строк t таких, что они имеют хотя бы 3 непересекающихся вхождения в строку s.