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

Бор, Ахо-Корасик, суфдерево

15 ноября

Собрано 20 декабря 2021 г. в 09:18

Содержание

Бор, 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 $\to$ SA**

   По суффиксному дереву постройте суффиксный массив с `LCP`.

3. `(*)` **Хитрый поиск**

   Дан словарь, постройте структуру, чтобы быстро отвечать на запросы

   a) `get(s)` – число строк в словаре, которые начинаются с s, заканчиваются на $\overleftarrow{s}$.
   b) `get(s,t)` – число строк в словаре, которые начинаются с s, заканчиваются на t, $|s| = |t|$.
   c) `get(s,t)` – число строк в словаре, которые начинаются с s, заканчиваются на t.

4. **Словари offline**

   Даны словарь (конечное множество слов) и текст.

   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.

6. **Разбиение на словарные слова**

   Дан словарь слов суммарной длины L и текст T.

   a) Слова длины $\leqslant l$. 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 минимального количества **подстрок** словарных слов за $\mathcal{O}(\text{poly}(L) + |T|)$.

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. **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 $\text{path}\colon root_A \rightsquigarrow u$. $\mathcal{O}(|A| + |B|)$. Размер алфавита $\mathcal{O}(1)$.

11. **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 $|A|$.

12. (*) **∞, 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.

13. (*) **XOR** $\to \max$

Дан массив a длины n. Найдите пару $a_i, a_j\colon a_i \hat{} a_j = \max$.

a) $\mathcal{O}(n \log M)$. $M = \max(a_1, \ldots, a_n)$.
b) $\mathcal{O}(\text{sort} + n)$

14. (*) **XOR** $\geqslant k$

Дан массив a длины n и число k. Длина чисел в массиве $\mathcal{O}(1)$. За время $\mathcal{O}(n)$ посчитайте количество

a) пар индексов таких, что побитовый `XOR` элементов по этим индексам $\geqslant k$.
b) отрезков последовательности, побитовый `XOR` всех чисел из которых $\geqslant k$.

15. (*) **Окно и стек**

Найти «количество различных подстрок» с помощью суффиксного массива.

Разбор задач практики

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

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

   Осталось придумать f: легко посчитать и сложно поддерлать. $f(x) = P^x \bmod M$.

   Заметим, что $\forall$`x` `x+=`Δ`x` $\Leftrightarrow$ $f(x)$`*=`$P^{\Delta\texttt{x}} \bmod M$

2. **ST $\to$ SA**

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

3. **Хитрый поиск**

   a) `get(s)` – бор из `LCP`$(w, \overleftarrow{w})$. Спускаемся по s, смотрим число конечных вершин в поддереве.
   b) `get(s,t)` – бор из строк $w_1 w_n w_2 w_{n-1} w_3 w_{n-2} \dots w_n w_1$. Спускаемся попеременно по символам s и $\overleftarrow{t}$. Смотрим число конечных вершин в поддереве.
   c) `get(s,t)` – бор прямых строк, отдельно бор обратных строк. Конечные вершины помечены номерами строк.
   Спускаемся в первом боре по s, во втором по $\overleftarrow{t}$, получили два поддерева. Ответ – пересечение множеств пометок в этих поддеревьях. В каждом дереве все пометки различны, умеем решать такую задачу 2D-запросом.

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.

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**

   a) **Число подстрок.** В боре это число вершин. В сжатом боре это сумма длин ребер.
   b) **Рефрен.** 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: $\min(R[v]-L[v], \text{depth}[v]) \to \max$. 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, храним min и max суффиксы в поддереве.
   e) $k \leqslant 64$ **строк.** Для каждой вершины считаем битмаску номеров строк, суффиксы которых есть в ее поддереве. Ищем самую глубокую, в поддереве которой есть все k.

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

9. **Бажный Укконен**

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

10. **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]`.

11. **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]]`.

12. `(*)` **∞, 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.

13. `(*)` **XOR** $\to \max$

    a) Числа – битовые строки длины **ровно** $\log M$.
    Строим на них бор, старшие биты ближе к корню.

Переберём a_i. Будем спускаться по бору, стараясь получить a_j с максимальным $a_i \hat{} a_j$: если есть ребро по биту, не равному соответствующему биту в a_i, спускаемся по нему.

b) $\mathcal{O}(\text{sort} + n)$. Сначала заметим, что можно не перебирать a_i, а найти оба элемента параллельным спуском по бору: идем по разным битам, если можем.
Далее, размер сжатого бора $\mathcal{O}(n)$. Имея сжатый бор, можно за $\mathcal{O}(n)$ найти ответ.
Сортируем массив и считаем LCP соседних. LCP можно посчитать битовыми операциями и предпосчитанными логарифмами степеней двоек.
По сортированному массиву и LCP умеем строить сжатый бор за $\mathcal{O}(n)$.

14. (*) **XOR $\geqslant k$**

a) Строим бор на числах массива как битовых строках равной длины.
Перебираем a_i, считаем число пар с ним. Для этого спускаемся по бору туда, где `XOR` даст результат, как в k. Если спустились по 0, прибавляем к ответу число листьев поддерева, куда ведёт ребро по 1.

b) `XOR` подотрезка $[i, j]$ – это $\text{pref}_i \hat{} \text{pref}_{j+1}$, где pref_i – `XOR` на префиксе. Решаем предыдущий пункт для pref.

15. (*) **Окно и стек**

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.

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. **(2) LowerBound**

   Напишите код на C++ структуры данных с интрефейсом

   a) `void add(const vector<int> &s);` – добавить строку.
   b) `vector<int> lowerbound(const vector<int> &s);` – найти строку лексикографически минимальную среди больше либо равных s.

   Размер алфавита $\mathcal{O}(1)$. За основу возьмите следующий бор:

```
struct Vertex {
    static const int ALPHABET = 26;
    vector<int> next;
    bool isEnd;
    Vertex() : next(ALPHABET, -1), isEnd(0) { }
}
vector<Vertex> trie(1); // бор из одной вершины
int root = 0;
```

5. **(3) Ключевые подстроки**

   Дан набор строк s_i. Для каждой s_i найдите min по длине подстроку, которая не встречается в других.

6. **(3) Уникальные суффиксы**

   Два запроса:

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

7. **(3) k-я общая подстрока**

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

   (+1) за произвольный размер алфавита.

3.2. Дополнительная часть

1. **(1) XOR-3**

   Дана массив натуральных чисел a длины $n \leqslant 10^6$. Найдите отрезок a, побитовый `XOR` всех чисел из которого максимален. Полный балл получит решение за время $\mathcal{O}(n + sort)$.

   На дополнительный балл можно придумать чистое $\mathcal{O}(n)$.

2. **(3) Поиск по отрезку словаря**

   Дан словарь $s_1, s_2, \ldots, s_n$. Отвечать в offline 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.

3. **(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).

4. **(3) Три вхождения**

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