SPb HSE, ПАДИИ, 1 курс, зима 2024/25
Практика по алгоритмам #21

Бор

27 февраля

Собрано 12 марта 2025 г. в 17:50

Содержание

Бор

1. **Бор**

   Рассмотрим строки $S = \{s_1, s_2, \ldots, s_n\}$ над алфавитом Σ.
   Бором для S называется min корневое дерево, из каждой вершины которого по каждому символу Σ исходит не более одного ребра, и $\forall i$ s_i является корректным путём от корня.

   a) Как хранить рёбра бора? Минимизируйте время/память, максимизируйте удобство.
   b) Сделайте на основе бора `unordered_set<string>`.
   c) Сделайте на основе бора `unordered_map<string, int>`.
   d) Сделайте на основе бора `set<string>`.
   e) Что сделать, чтобы все строки заканчивались только в листьях?

2. **Сортировка строк**

   Дан набор из n строк суммарной длины L над алфавитом A.

   a) Отсортируйте за время $\mathcal{O}(L \log |A|)$.
   b) Покажите, что $\nexists$ алгоритма сортировки строк за $\mathcal{O}(L)$.

3. **Хаффман**

   Есть замечательный способ сжатия текста: каждому символу алфавита сопоставить последовательность бит (код) не обязательно длины 8, а произвольной. Теперь закодированный текст – последовательность бит, полученная конкатенацией битовых кодов символов.

   Каким свойством должны обладать коды символов, чтобы точно можно было однозначно раскодировать? Придумайте, как раскодировать за $\mathcal{O}(n)$, где n – битовая длина закодированного текста.

   Как выбрать символам оптимальные коды? Понятно, что частым символам коды покороче, редким символам коды подлиннее, но как именно? Придумайте, доказывать не нужно.

4. **k-ая строка**

   Структура для строк: `add`, `del`, вернуть k-ую.

5. **T9**

   Дан словарь. Понимать быстро по строке s, сколько словарных слов на неё начинается.

   А как вывести самое лучшее слово, начинающееся на s?

   А три самых лучших?

   Как перестроить структуру, если вес (мера хорошести) слова изменился?

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) 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) Общая подстрока двух строк.

8. **Бор – это не только про строки**

   Нужна структура данных, умеющая `add(x)`, `del(x)`, `get-kth(k)` для целых 32-битных чисел.

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

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

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

10. `(*)` **Сортировка строк**

    a) `(*)` Отсортируйте за время $\mathcal{O}(L + |A|)$.
    b) `(*)` За сколько `QuickSort` сортирует строки?

Разбор задач практики

1. **Бор**
   a) **Хранение.** `v.next[c]` дает ребро из вершины `v` по символу `c`.
   `next` может быть массивом, `map`, `unordered_map`.
   Если массив, то $\mathcal{O}(L|\Sigma|)$ памяти, иначе $\mathcal{O}(L)$, L – суммарная длина строк.
   b) `unordered_set<string>`. При добавлении строки создаем все нужные вершины и ребра, если их еще нет. В вершинах, где кончается строка, ставим пометку, что они конечные.
   c) `unordered_map<string, int>`. Дополнительное поле в конечных вершинах.
   d) `set<string>`. `v.next` – массив или `map`. Тогда можно перебирать строки в сортированном порядке и делать `lower_bound`.
   e) Чтобы строки заканчивались только в листьях, добавим в конец каждой символ, не встречающийся в строках.

2. **Сортировка строк**
   a) $\mathcal{O}(L \log |A|)$. Строим бор, в вершинах `map<char,int>`. `dfs` по бору, из вершины идем в детей в порядке возрастания символа.
   b) Если очень большой алфавит и все строки длины 1, то сортировка строк не проще сортировки L чисел.
   c) $\mathcal{O}(L+|\Sigma|)$. Отсортируем поразрядно пары $\langle j, s_i[j]\rangle$, т.е. «позиция в строке, символ». Также про каждую пару помним, из какой она строки.
   Теперь можно класть в бор сразу упорядоченные ребра. Берем пару, смотрим, в какую вершину бора пришли по соответствующей строке. Добавляем в нее соответствующее ребро, либо оно там уже есть, тогда это ровно последнее добавленное.

3. **Хаффман**

   Свойство: ни какой код не является префиксом другого. Распаковка: сложили все коды в бор, кушаем биты запакованного текста, спускаемся по бору до листа, в листе написан символ. Запаковка: заменим пару букв с минимальными частотами x и y на вершину $x+y$ из которой торчит ребро по нулю в x и по 1 в y. Так получили сразу бор кодов.

4. **k-ая строка**

   Нужно поддерживать в боре «сколько строк заканчивается в поддереве», чтобы понимать по какому ребру спускаться к k-й.

5. **T9**

   Засунуть строки в бор, насчитать «сколько строк заканчивается в поддереве».

   Ответ на задачу: спустить по строк, посмотреть, что хранится в вершине бора.

   А ещё в каждом поддереве хранить «max вес в поддереве».

   Или, если просят, в каждом поддереве хранить сразу «три max веса в поддереве».

   Если вес поменялся, ок, пересчитаем максимумы на пути до корня.

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 $s_1\#s_2\#\ldots\#s_n$ 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.

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.* count(s) – число суффиксов, кончающихся в поддереве, если спуститься по s. Заметим, что достаточно смотреть на строки, кончающиеся в вершинах.
c) Строим суффиксное дерево от строки `s#t`, считаем динамику «есть ли конец s-суффикса в поддереве» и «есть ли конец t-суффикса».

8. **Бор – это не только про строки**

Число = битовая строка длины 32 $\Rightarrow$ задача решается бором, страшие биты ближе к корню.

9. `(*)` **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 $\mathcal{O}(n)$.

Домашнее задание

1. **(2) T9**

   Дан набор n строк s_i. Для каждой s_i найдите min по длине префикс, который не является префиксом других строк. Время $\mathcal{O}(\sum |s_i|)$. Практическое значение: именно столько нажатий нужно до того, как автодополнение автоматически может подставить нужное слово.

   (+0.5), если у вас получится $\mathcal{O}(n)$ доппамяти.

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

   По суффиксному дереву постройте суффиксный массив с `LCP` за $\mathcal{O}(n)$. `LCP`$[i]$ = наибольший общий префикс i-го и $(i{+}1)$-го по порядку суффикса в суффиксном массиве.

3.1. Дополнительная часть

1. **(2) SA $\to$ ST**

   По суффиксному массиву с `LCP` за $\mathcal{O}(n)$ постройте суффиксное дерево.

2. **(2) Ключевые подстроки**

   Дан набор строк s_i. Для каждой s_i найдите минимальную по длине подстроку, которая не встречается в других s_j. Количество и суммарная длина строк до 10^5.