Первый курс, весенний семестр

Практика по алгоритмам #10

Строки: бор, Ахо-Корасик

Contents

1 Новые задачи **2**

2 Домашнее задание **3**

2.1 Обязательная часть 3

2.2 Дополнительная часть 4

1 Новые задачи

1. **Quicksort.**
   Оцените время работы `QuickSort` для строк с наивной функцией сравнения.
2. **Словари offline.**
   Даны словарь (конечное множество слов) и текст.
   a) Для каждого слова из словаря определите, входит ли оно как подстрока в текст.
   b) Теперь для каждого слова из словаря определите количество вхождений в текст.
3. **Словари online.**
   Снова даны словарь (конечное множество слов) и текст. Теперь нужно уметь обновлять ответ `online` при добавлении символов в конец текста.
   a) Пересчитайте суммарное число вхождений слов из словаря в текст за $\mathcal{O}(1)$.
   b) Поддерживайте множество всех вхождений слов из словаря в текст. Пересчёт за время $\mathcal{O}(1+|\Delta A|)$, где ΔA – прирощение ответа после добавления очередного символа.
4. **Сортировка строк.**
   Дан набор строк суммарной длины n над алфавитом Σ .
   a) Отсортируйте за время $\mathcal{O}(n \log |\Sigma|)$.
   b) Покажите, что за $\mathcal{O}(n)$ нельзя.
5. **Разбиение на словарные слова.**
   Дан словарь слов суммарной длины L и текст T. Длины слов в словаре не более l. Представьте текст в виде конкатенации минимального количества словарных слов за время $\mathcal{O}(L + l|T|)$. Слова можно использовать более одного раза.
6. **Разбиение на подстроки словарных слов.**
   Дан словарь слов суммарной длины L и текст T. Представьте текст в виде конкатенации минимального количества **подстрок** словарных слов за время $\mathcal{O}(Poly(L) + |T|)$.
7. **Один бор хорошо, а два — лучше!**
   Даны два множества строк, хранящиеся в виде боров A и B (с $|A|$ и $|B|$ вершинами соответственно). Найдите для каждой вершины $u \in A$ самую глубокую вершину B, путь до которой соответствует суффиксу $path(A)$. $\mathcal{O}(|A| + |B|)$.
8. **XOR-1** $\to \max$
   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) (*) $\mathcal{O}(n)$
9. **XOR-2** $\geq k$
   Дан массив a длины n и число k. За время $\mathcal{O}(n)$ посчитайте количество
   a) пар индексов таких, что побитовый `xor` элементов по этим индексам $\geq k$.
   b) отрезков последовательности, побитовый `xor` всех чисел из которых $\geq k$.
10. ∞
    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. **Regexp**
    Дан паттерн — строка длины n, состоящая из букв и знаков “?”. Знаков “?” не более k. Вместо каждого знака “?” можно подставить любую букву. Предложите практически эффективный алгоритм поиска паттерна в тексте.

2 Домашнее задание

2.1 Обязательная часть

1. **(2) Снова словари.**
   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. Нужно вернуть вершину бора v, от которой строку s можно отложить вниз. Размер алфавита $\mathcal{O}(1)$. Время $\mathcal{O}(|A| + |s|)$.

3. **(3) LowerBound.**
   Напишите псевдокод 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 больше либо равную данной среди уже добавленных.

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

```
struct Vertex {
  static const int ALPHABET = 26;
  int next[ALPHABET];
  bool isEnd;
  Vertex() {
    memset(next, -1, sizeof(next));
    isEnd = 0;
  }
};
vector<Vertex> trie(1);
int root = 0;
```

4. **(2) XOR-3**
   Дана массив a длины n. За время $\mathcal{O}(n)$ найдите отрезок последовательности, побитовый `xor` всех чисел из которого максимален.

5. **(3) Количество строк.**
   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. Можно получить +1 допбалл за $\mathcal{O}(L)$ памяти.

6. **(3) Динамический словарь.**
   В словаре теперь могут добавляются и удаляются слова. Необходимо в `online` научится 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).

7. **(2) Дополним строку до палиндрома!**
   Дана строка s, нужно за $\mathcal{O}(|s|)$ найти такую минимальную строку t, что st – палиндром.

8. **(2) Разбить строку на палиндромы**
   Дана строка s, нужно за $\mathcal{O}(|s|^2)$ представить её в виде конкатенации минимального числа палиндромов.

9. (3) **Разбить строку на 2 почти палиндрома**
   Проверить за $\mathcal{O}(|s|)$, можно ли строку s разбить на два почти палиндрома. Почти палиндром – строка, в которой можно заменить не более одного символа, чтобы получился палиндром.

2.2 Дополнительная часть

1. (4) **РОИ-2004.Задача-2.**
   Даны n словарных слов и m слов текста. Суммарная длина всех слов L. Слова похожи, если можно из каждого удалить не более одной буквы, чтобы они стали равны. Найдите для каждого слова текста:

   a) какое-нибудь похожее слово словаря.
   b) кол-во похожих слов в словаре.

   Количество баллов зависит от асимптотики. Существует решение за $\mathcal{O}(L)$
2. (4) **Подпалиндромы на отрезках.**
   Дана строка s и q запросов "самый длинный подпалиндром на отрезке строки $[l..r]$".
   Решите за $\mathcal{O}((|s|+q)PolyLog)$.
3. (5) **Разбить строку на 3 палиндрома.**
   Дана строка s, нужно за $\mathcal{O}(|s|)$ представить её в виде конкатенации 3 палиндромов.
4. (7) **Разбить строку на 31 палиндром.**
   `<link to timus>`