Второй курс, осенний семестр 2021/22
Практика по алгоритмам #13

Финал близок
13 декабря

Собрано 13 декабря 2021 г. в 21:39

Содержание

Финал близок

1. **НОП за $\mathcal{O}(n^2/\log^2 n)$**

   Даны строки s и t над алфавитом $\{0,1\}$, $|s|,|t| \leqslant n$.
   Наша цель — найти их наибольшую общую **подпоследовательность** за $\mathcal{O}(n^2/\log^2 n)$.

   a) Опишите динамическое программирование `dp` за $\mathcal{O}(n^2)$.
   b) Постройте $\{0,1\}$-матрицу/матрицы, по которым однозначно восстанавливается `dp`.
   c) Разобьём матрицы на блоки $k \times k$ (k выберем в конце). Хотим, чтобы любой блок однозначно задавался небольшим числом бит m. Что это за биты?
   d) Опишите предподсчёт для всех типов блоков за $\mathcal{O}(2^m \cdot \mathrm{poly}(k))$.
   Что будет храниться в предподсчёте?
   e) Оцените итоговую асимптотику. Выберите правильное k.
   f) `(*)` Что делать, если алфавит большой (например, 26)?

2. **Левенштейн**

   Найдите расстояние Левенштейна за $\mathcal{O}(n^2/\log^2 n)$.

3. **RMQ за $\mathcal{O}(1)$**

   Разделим массив на куски длины $w = 64$. На полученных $\frac{n}{w}$ кусках насчитаем SparseTable.
   Внутри каждого куска пройдёмся со стеком, поддерживая «стек локальных минимумов»:

   ```
   add(i): { while (a[s.top()] >= a[i]) s.pop(); } s.push(i);
   ```

   Для каждого префикса $[0, R)$ можно вместе со стеком поддерживать и `f[R]`
   «маску единичных бит»: `mask ^= 1 << s.top()` и `mask ^= 1 << i`, `f[R] = mask`.

   Придумайте, как, пользуясь таким предподсчётом, за $\mathcal{O}(1)$ возвращать минимум на отрезке.
   Сравните полученный метод с решением задачи RMQ алгоритмом Фараха-Колтона-Бендера.

4. **4-русских для умножения матриц**

   На лекции мы изучили умножение матриц над $\mathbb{F}_2$. А как умножить булевы матрицы (операции в булевом полукольце)? Сложение: `OR`, умножение: `AND`.

5. **Транзитивное замыкание за $\mathcal{O}(\log n)$ умножений**

   Построение транзитивного замыкания графа связано с возведением матрицы смежности в степень. За сколько мы теперь умеем находить транзитивное замыкание?

6. **Транзитивное замыкание за $\mathcal{O}($`MatrixMul`$)$**

   Сожмите компоненты сильной связности, найдите топсорт конденсации. Примените метод разделяй и властвуй на топсорте, чтобы получить время работы $T(n) = 2T(\frac{n}{2}) + 2 \cdot$ `MatrixMul`.

7. **Инкрементальное транзитивное замыкание**

   Пересчитайте матрицу достижимости после добавления ребра $a \to b$.

   a) За $\mathcal{O}(n^2)$
   b) За $\mathcal{O}(n^2/w)$
   c) За $\mathcal{O}(n)$ с учётом амортизации: за всё время добавится $q \leqslant n^2$ рёбер,
   нужно, чтобы ваш алгоритм работал за $(n^2+q) \cdot \mathcal{O}(n)$.
   d) `(*)` За $\mathcal{O}(\frac{n}{\log n})$ с учётом амортизации, за $(n^2+q) \cdot \mathcal{O}(\frac{n}{\log n})$.

8. (*) **Транспонирование матрицы**

Придумайте, как хранить матрицу над $\mathbb{F}_2$, чтобы доступ к элементу $[i, j]$ на чтение и запись был за $\mathcal{O}(1)$, а транспонировать матрицу можно было бы за $o(n^2)$.

Разбор задач практики

1. **НОП за $\mathcal{O}(n^2/\log^2 n)$**
   Конспект прошлых лет — часть 14.6, страница 62/69.
   a) $f[i,j]$ – длина НОП $s[0{:}i)$ и $t[0{:}j)$
   b) $x[i,j] = f[i,j{+}1] - f[i,j]$, $y[i,j] = f[i{+}1,j] - f[i,j]$.
   На самом деле сейчас нам хватает любой одной из них. Скоро понадобятся обе.
   c) На картинке (матрица, разбитая на блоки + строки, как оси координат) легко показать, что блок $f[i..i{+}k, j..j{+}k]$ зависит от 4 отрезков по k бит ($4k$ бит) – $s[i..i{+}k]$, $t[j..j{+}k]$, сжатая первая строка блока f, сжатый первый столбец блока f.
   d) Перебор числа $0 \leqslant m < 2^{4k}$. Раскодируем m в 4 массива по k бит.
   Строим первый столбец и первую строку блока матрицы f. Считаем блок f в лоб за k^2.
   Сжимаем последний столбец и строку блока f в $2k$ бит, их и храним в $dp[m]$.
   e) Работает за $2^{4k}k^2 + (n/k)^2$. С ростом k, первое $\nearrow$, второе $\searrow$.
   $k = \frac{1}{2}\log n$ – чуть многовато $\Rightarrow$ возьмём с запасом $k = \frac{1}{4}\log n$, получим $\mathcal{O}(n^2/\log^2 n)$.

2. **Левенштейн**
   Также как НОП. Ровно также.

3. **RMQ за $\mathcal{O}(1)$**
   `mask[R]` = возрастающий стек минимумов на $[0..R)$
   `getMin [L,R) = lowerBit(mask[R] & ~(2^L-1))`
   `lowerBit(x) = table[(x - (x&(x-1))) % 67]` (совершенная хеш-функция для w=64)

4. **4-русских для умножения матриц**
   Всё работает также.
   Внутри динамики от 4 русских теперь `OR` `bitset`-ов вместо `XOR`-а `bitset`-ов.

5. **Транзитивное замыкание за $\mathcal{O}(\log n)$ умножений**
   Добавим в матрицу смежности петли. Возведём её в степень n. Получили ответ.

6. **Транзитивное замыкание за $\mathcal{O}$(`MatrixMul`)**
   Пусть левая половина A, правая половина B, матрица перехода из A в B — это C. Достижимость внутри A и B ($ans(A)$ и $ans(B)$) получаются двумя рекурсивными вызовами. Из B в A нельзя попасть т.к. граф ацикличен. Достижимость $A \to B$ получается как $ans(A)\cdot C\cdot ans(B)$

7. **Инкрементальное транзитивное замыкание**
   a) За $\mathcal{O}(n^2)$: `for x for y : d[x,y] |= d[x,a]&d[b,y]`
   b) За $\mathcal{O}(n^2/w)$: `for x if d[x,a] : d[x] |= d[b]` (bitset!)
   c) За $\mathcal{O}(n)$. Найдём за $\mathcal{O}(n)$ все x: $d[x,a]{=}1 \wedge d[x,b] = 0$. Для каждого такого x появится хотя бы одна единица $d[x,b]$. Найдём за $\mathcal{O}(n)$ все y: $d[b,y]{=}1 \wedge d[a,y] = 0$. Для каждого такого y появится хотя бы одна единица $d[a,y]$. Задача: $\forall x, y$ проставить единицы в $d[x,y]$.
   Время работы: $n + |X|\cdot|Y|$. Число новых единиц $\geqslant |X| + |Y|$.
   d) За $\mathcal{O}(n/w)$. Будем поддерживать и d, и d^T,
   тогда `bitset` всех x и y можно найти за $\frac{n}{w} + |X| + |Y|$.
   $\forall x$ найдём $Z_x = d[x] \cap Y$ и переберём все $z \in Z_x$ за $\mathcal{O}(\frac{n}{w})$.
   $\forall x, \forall z \in Z_x$ проставим $d[x,z] = 1$, $d^T[z,x] = 1$.

8. (*) **Транспонирование матрицы**

Разобьём исходную матрицу на куски $\sqrt{\log n} \times \sqrt{\log n}$.
Будем хранить матрицу, как массив $\frac{k}{\sqrt{\log n}} \times \frac{k}{\sqrt{\log n}}$ кусков, каждый кусок — один `int`.
Транспонирование работает за $\frac{n^2}{\log n}$ транспонирований куска. Транспонирование куска — $\sqrt{\log n}$ обращений к таблице прекалка размера $2^{\sqrt{\log n}}$. Итого $\frac{n^2}{\sqrt{\log n}}$.