Домашняя работа

Котов Артем, МОиАД2020

3 октября 2020 г.

Содержание

Замечание. *Почти везде, вроде как, я буду использовать массив d для отслеживания некоторой характеристики состояний, по нему же будет строится динамика*

Task 1

Условие: Дана строка s длины n. Для каждой пары (i, j) найти длину максимального общего префикса i-го и j-го суффиксов строки s. $\mathcal{O}(n^2)$.

Решение.

Введем d_{ij} — длина максимальной общей префиксной подстроки i-ого и j-ого суффиксов. Естественно ограничить значения для этого массива как $d_{in} = \begin{cases} 0, \text{if } s_i \neq s_n \\ 1, \text{otherwise} \end{cases}$, аналогично d_{nj}, также $d_{nn} = 1$.

Построим динамику (с учетом, что нельзя пробить потолок, то есть $i + 1$,$j + 1 \leq n$):

$$d_{ij} = \begin{cases} 0, \text{if } s_i \neq s_j \\ 1 + d_{i+1,j+1} \end{cases}$$

Нам надо будет так пробежаться по всем i и j от $n-1$ до 1 и проделать какие-то константные операции, то есть будет $\mathcal{O}(n^2)$. Массив d и будет в данном случае ответом.

Update: Из условного псевдокодика, как мне кажется, будет нагляднее пробежка по всему d_{ij}

```
# init
d[n,n] = 1
for i in range(1,n):
    if s[i] != s[n]:
        d[i,n] = d[n,i] = 0
    else:
        d[i,n] = d[n,i] = 1

# dynamic
for i in range(n-1, 1, -1):
    for j in range(n-1, 1, -1):
        if s[i] != s[j]:
            d[i,j] = 0
        else:
            d[i,j] = 1 + d[i+1, j+1]
```

∎

Task 2

Условие: Дан набор нечестных монеток с вероятностью выпадения орла $p1, p2, \ldots, pn$. Требуется посчитать вероятность выпадения ровно k орлов за $\mathcal{O}(n \cdot k)$. Операции над числами считать выполнимыми за $\mathcal{O}(1)$.

Решение.

Рассмотрим следующий массив d_{kn}, означающий вероятность выбросить k орлов на n-ом броске. Например, $d_{01} = 1 - p_1$, $d_{12} = p_1(1 - p_2) + p_2(1 - p_1)$ Можем задаться вопросом, как эта вероятность связана с предыдущими бросками: либо на $n - 1$ броске уже было выброшено k орлов, и тогда нам надо не выбросить орла не n-ом броске, либо на $n - 1$ броске было выброшено $k - 1$ орел (если меньше, то заведомо на последнем броске невозможно получить достаточно количество орлов), тогда нам надо выбросить еще одного орла, то есть динамика такая:

$$d_{kn} = (1 - p_n)d_{k,n-1} + p_n d_{k-1,n-1}$$

то есть мы пробегаемся по всем n k раз (ну что-то порядка этого числа раз, то есть $\mathcal{O}(nk)$ раз), делая какие-то константные действия, итоговая сложность будет $\mathcal{O}(nk)$, в ответ уйдет значение d_{kn}

Update: Граничные условия можно поставить следующие: $d_{00} = 1$, $d_{kk} = \prod\limits_{i=1}^{k} p_i \ \forall k \in [1,n]$, еще можно взять кусочки из примера.

■

Task 3

Условие: Дан массив из n целых чисел и число d. Найти подпоследовательность максимальной длины с условием, что соседние элементы в ней должны отличаться не более чем на d за $\mathcal{O}(n^2)$.

Решение.

Поступим как с поиском возрастающей последовательности, но с другим условием (в поиске возрастающей последовательности мы смотрели, что $a_i < a_j$), а именно $|a_j - a_i| \leq d$. То есть составим массив d_i — длина самой длинной последовательности, разница между соседними элементами которой не больше d, при этом эта последовательность заканчивается на a_i-ом элементе исходного массива. Динамика: $d_i = 1 + \max\limits_{1<j<i}(d_j)$ при условии, что $|a_j - a_i| \leq d$. Если такого j нет, то $d_i = 1$. Это делается за $\mathcal{O}(n^2)$. ~~В ответ выводим $\max\limits_i d_i$, что делается еще за $\mathcal{O}(n)$ но это уже не важно.~~

Update: Значит, нам надо будет запоминать последовательность, например, запоминать для каждого i из какого элемента j мы пришли (пусть это будет массив `previous[i]`), тогда, найдя максимум d_i, с помощью таких указателей на предыдущий элемент мы сможем восстановить саму последовательность.

$\blacksquare$

Task 4

Условие: Дан массив из n целых чисел, число d и число k. Найти подпоследовательность длины k с максимальной суммой элементов при условии, что соседние элементы в ней должны отличаться не более чем на d. $\mathcal{O}(n^2k)$.

Решение.

Заведем массив d_{ij} — максимальная сумма такой последовательности $1 < j < k$ штук элементов от 0-ого до i-ого исходного массива, что она заканчивается a_i-ым элементом исходного массива. Естественным ограничением будет $d_{i1} = a_i \; \forall i \in [1, n]$, так как только сам элемент и является такой последовательностью длины 1. Теперь надо поступить почти также как с поиском возрастающей последовательности: $d_{ij} = \max\limits_{l<i}(d_{ij}, d_{l,j-1} + a_i) \; \forall j \in [2, k]$, то есть если существует такая последовательность длины $j-1$, оканчивающаяся раньше i, что сумма ее значения и текущего нового элемента больше, чем текущее максимальное значение для последовательности, заканчивающейся на текущем элементе, то стоит обновить значение для текущего элемента. Так как тут добавилась еще и пробежка по всем $j \in [2, k]$, то сложность будет уже $\mathcal{O}(n^2k)$. ~~В ответ надо будет выдать $\max\limits_i(d_{ik})$~~
Update: Видимо, можно поступить как и в предыдущей задачи: надо будет сохранять указатели на предыдущие элементы последовательности. Вообще, казалось бы, надо для каждого k это сделать, но мы все равно интересуемся каким-то конкретным k, так что все должно быть ок даже по памяти (то есть это может быть один массив, который с ростом k тоже растет). То есть опять таки, найдя $\max\limits_i(d_{ik})$ по указателям на предыдущий элемент восстановим последовательность.

$\blacksquare$

Task 6

Условие: Клетки поля $n \times 5$ покрашены в чёрный и белый цвета. Будем называть получившийся узор красивым, если он не содержит одноцветного квадрата 2×2. Вычислите число красивых узоров по модулю небольшого простого числа за время $\mathcal{O}(\log n)$.

Решение.

Попробуем свести это дело к какому-то рекуррентному соотношению. Рассмотрим столбцы длины 5 (то есть наша плитка будет расти слева направо). Всего существует 32 раскраски таких столбцов. Заведем массив a_{ik} — количество раскрасок k столбцов, заканчивающихся каким-то конкретным i-ым столбцом. Также заведем еще массив d_{ij} — говорит нам, можем ли мы поставить раскрашенный столбец i рядом с раскрашенным столбцом j (0, если не можем, 1, если можем), чтобы узор оставался красивым. Естественное ограничение $a_{i1} = 1$, так как существует ровно одна какая-то конкретная раскраска одного столбца, всегда же $1 \le i \le 32$.

Теперь зададимся вопросом, как связано количество раскрасок a_{ik} с предыдущими раскрасками, то есть какова динамика? Рассмотрим $a_{ik} = \sum\limits_{j=1}^{32} a_{j,k-1} d_{ji}$, то есть мы перебираем всевозможные раскраски последнего, j-ого столбца и проверяем, можем ли мы поставить текущий новый i-ый столбец в такую комбинацию, ну и естественно просуммировать подходящие варианты красивых раскрасок.

Теперь отнесемся к i у a как к индексу некоторого вектора, тогда динамику можно переписать так $\boldsymbol{a}_k = \mathcal{D}\boldsymbol{a}_{k-1}$, где матрица $\mathcal{D}$ — матрица d, причем симметричная. То есть мы получили хороший линейный рекуррент в матричной форме.

Определим начальные условия, то есть $\boldsymbol{a}_1$. Как раньше уже упоминалось, у нас есть ограничение $a_{i1} = 1$, тогда $\boldsymbol{a}_1 = (1, \ldots, 1)^T$, теперь мы можем окончательно сформулировать нашу задачу: $\boldsymbol{a}_n = \mathcal{D}^n \boldsymbol{a}_1$. Возводить матрицу в степень мы умеем по следующей схеме: $\mathcal{D}^n = \mathcal{D}^{n/2}\mathcal{D}^{n/2} = \mathcal{D}^{n/4}\mathcal{D}^{n/4}\mathcal{D}^{n/4}\mathcal{D}^{n/4}$, то есть за $\mathcal{O}(\log n)$. ■