Можно брать отсюда задачи - <https://medium.com/techie-delight/top-50-dynamic-programming-practice-problems-4208fed71aa3>

1. Найдите подпоследовательность заданной последовательности такую, чтобы сумма подпоследовательности была как можно больше, а элементы подпоследовательности отсортированы в возрастающем порядке. Эта подпоследовательность не обязательно непрерывна или уникальна.

Пример: {0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11} => {8, 12, 14}.

**Решение:** [**https://www.techiedelight.com/increasing-subsequence-with-maximum-sum/**](https://www.techiedelight.com/increasing-subsequence-with-maximum-sum/)

Базово - использовать рекурсию. Для каждого элемента есть две возможности:

* Включить текущий элемент, если он больше, чем предыдущий элемент, и повторить для остальных элементов.
* Исключить текущий элемент и повторить для остальных элементов.

Базовым случаем рекурсии будет отсутствие элементов. Экспонента по времени.

Если будем запоминать - будет нисходящее динамическое программирование

Мы также можем решить эту проблему восходящим способом. При подходе «снизу вверх» мы сначала решаем меньшие подзадачи, а затем на их основе решаем более крупные подзадачи. Проходимся слева направо и для каждого элемента проверяем все предыдущие суммы, и если элемент больше, чем предыдущий, то среди таких находим максимальную сумму (базово - сумма равна самому элементу)

Сложность - квадрат, место - линия.

1. Найти подпоследовательность заданной последовательности, в которой элементы подпоследовательности отсортированы в порядке от низшего к высшему, и в которой подпоследовательность является максимально возможной. Эта подпоследовательность не обязательно непрерывна или уникальна.

Пример: [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15] -> [0, 2, 6, 9 , 11, 15]. (Самая длинная возрастающая подпоследовательность в этом примере не уникальна, например, [0, 4, 6, 9, 11, 15] или [0, 4, 6, 9, 13, 15])

**Решение:** [**https://www.techiedelight.com/longest-increasing-subsequence-using-dynamic-programming/**](https://www.techiedelight.com/longest-increasing-subsequence-using-dynamic-programming/)

Аналогично предыдущей задаче, только считаем количество элементов, а не сумму

1. Найти количество n-значных двоичных чисел, не содержащих двух подряд идущих единиц.

Пример: n=5 -> 13 (00000, 00001, 00010, 00100, 00101, 01000, 01001, 01010, 10000, 10001, 10010, 10100, 10101)

**Решение:** <https://www.techiedelight.com/find-n-digit-binary-strings-without-consecutive-1s/>

Перебор всех экспоненциален, а нам бы хорошо просто хранить последний возможный знак числа. База - 0 и 1 подходят, то есть по 1 числу оканчиваются на 0 и на 1. Из двухзначных на 1 могут заканчиваться только те, что оканчивались на 0 для n-1 знака, а на 0 - любые. Несложно заметить, что получаем числа Фибоначчи =) (для двухзначных - 2 числа оканчиваются на 0 и одно на 1, для трехзначных - 3 и 2, для четырехзначных - 5 и 3, для пятизначных - 8 и 5 и т.д.)

1. Найти наименьшую общую суперпоследовательность, то есть такую последовательность, что обе последовательности являются ее подпоследовательностями (не обязательно подстрокой!)

Пример: ABCBDAB и BDCABA -> ABCBDCABA, ABDCABDAB или ABDCBDABA.

**Решение:** [**https://www.techiedelight.com/shortest-common-supersequence-introduction-scs-length/**](https://www.techiedelight.com/shortest-common-supersequence-introduction-scs-length/)

Если последние символы совпадают, то L(i-1, j-1)+1, иначе max(L(i-1, j), L(i, j-1)) + 1.

1. Найти наибольшую общую подстроку.

Пример: ABABC и BABCA -> BABC.

**Решение:** [**https://www.techiedelight.com/longest-common-substring-problem/**](https://www.techiedelight.com/longest-common-substring-problem/)

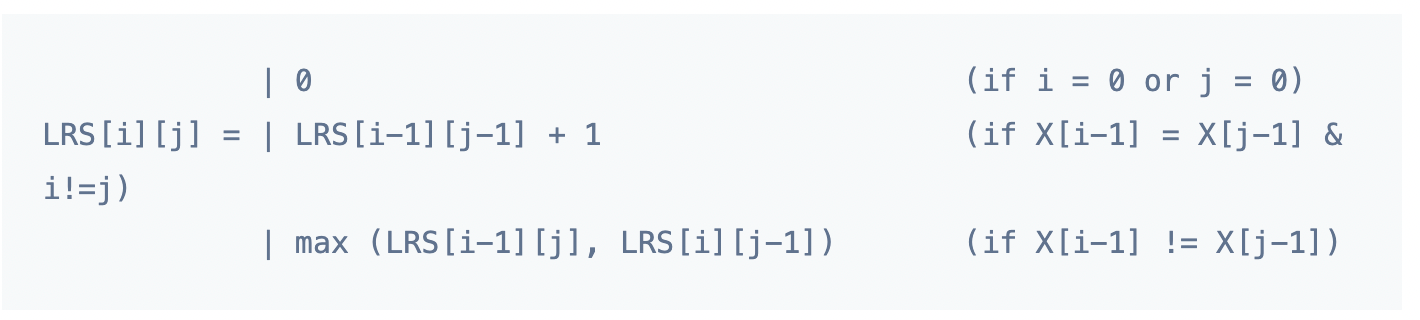
Матрица MxN, соответствующая символам в словах, правило обновления - если символы равны, то L[i][j] = L[i-1][j-1] + 1

1. Найти самую длинную повторяющуюся подпоследовательность (не обязательно подстроку!)

Пример: ATACTCGGA -> ATCG

**Решение:** [**https://www.techiedelight.com/longest-repeated-subsequence-problem/**](https://www.techiedelight.com/longest-repeated-subsequence-problem/)

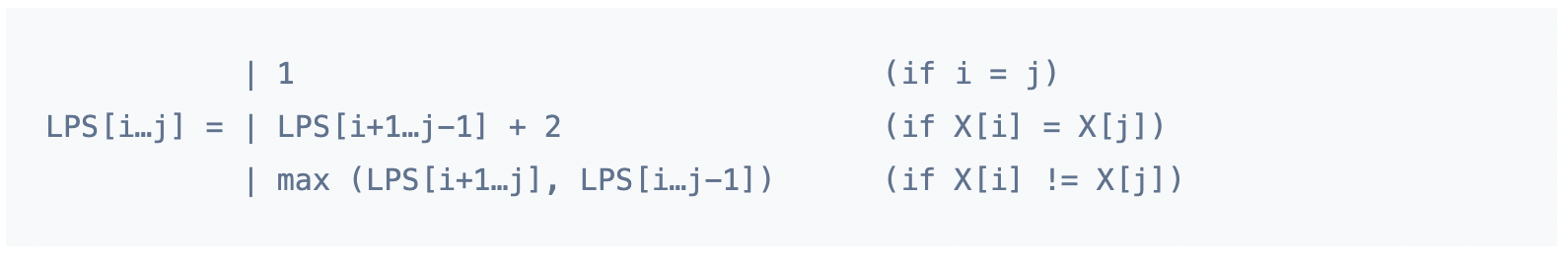
(i, j) пропускаем, так как символ не может входить сразу в две подпоследовательности - это будет незачет

****

1. Найти для данной строки длиннейшую подпоследовательность, которая является палиндромом

Пример: ABBDACAB -> BACAB

**Решение**: <https://www.techiedelight.com/longest-palindromic-subsequence-using-dynamic-programming/>



1. Дан набор чисел. Необходимо найти длину наибольшей чередующейся последовательности, то есть последовательности, в которой a1<a2>a3<a4… или аналогично начиная со знака >

Пример: [8, 9, 6, 4, 5, 7, 3, 2, 4] -> 6 (например, [8, 9, 4, 7, 2, 4])

**Решение:** <https://www.techiedelight.com/longest-alternating-subsequence/>

Задание с подвохом - его можно решить и в один проход за линейное время, решение есть в комментариях

1. Дана строка и дан набор слов. Необходимо определить, может ли строка быть разбита на слова, содержащиеся в словаре.

Пример:

d = ['self', 'th', 'is', 'famous', 'Word', 'break', 'b', 'r', 'e', 'a', 'k', 'br', 'bre', 'brea', 'ak', 'problem']

s = ‘Wordbreakproblem’

Вывод: True (Word break problem или Word brea k problem или Word br e ak problem или другие)

**Решение:** <https://www.techiedelight.com/word-break-problem/>

Для строки проверяем поочередно все префиксы на то, являются ли они словом из словаря, если да - рекурсивно проверяем остаток строки. С запоминанием, естественно. Квадрат по времени, линия по памяти (храним только нолики и единички для всех суффиксов, чтобы их не перепроверять)

1. Дана матрица MxN, заполненная целыми числами - стоимостью ячейки. Надо найти самый дешевый путь из верхнего левого угла (0, 0) в нижний правый (M-1, N-1). Перемещаться можно только вправо и вниз. В примере ниже путь выделен синим, а его стоимость равна 36.

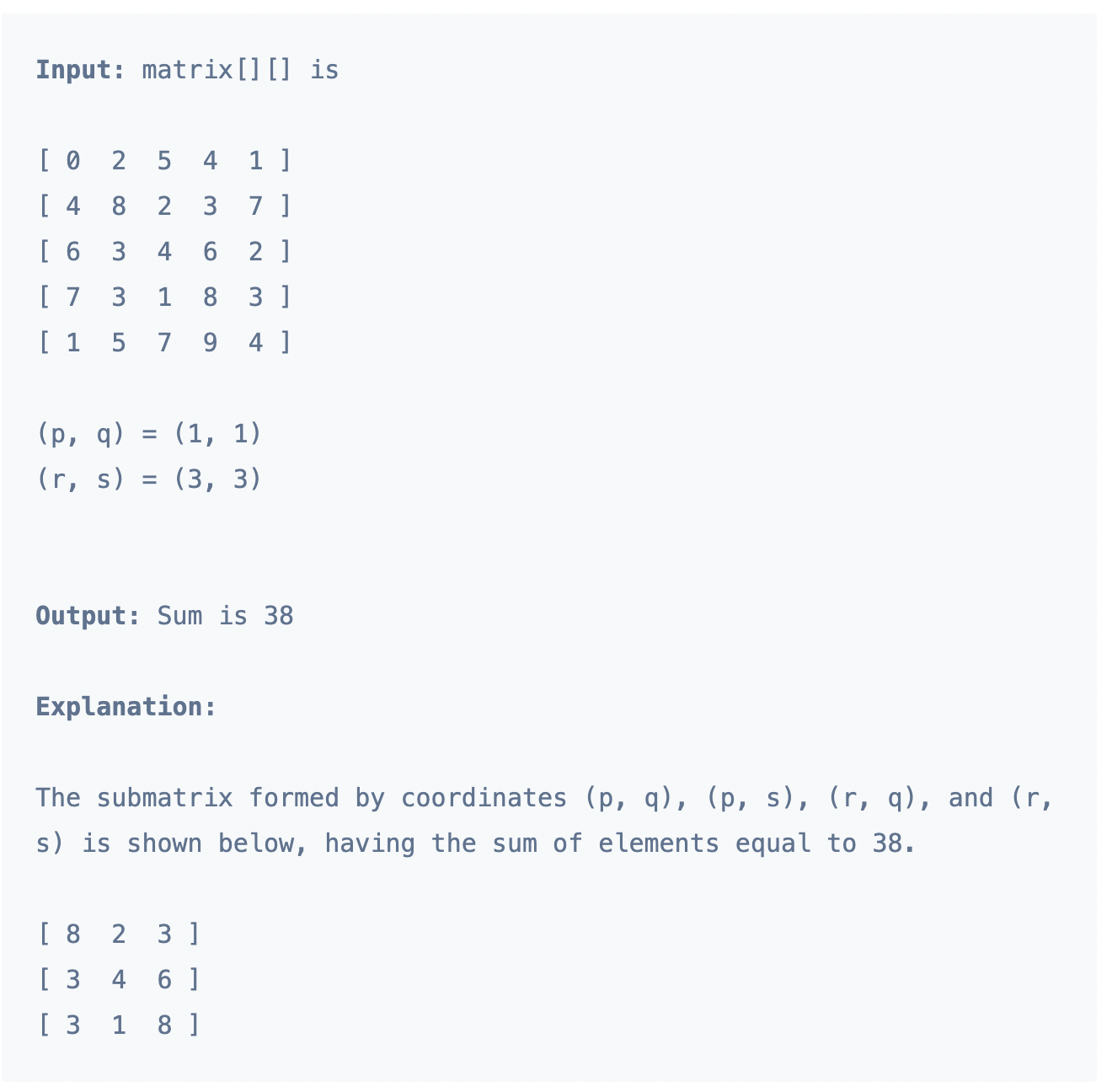
**A picture containing diagram

Description automatically generated**

**Решение:** [**https://www.techiedelight.com/find-minimum-cost-reach-last-cell-matrix-first-cell**](https://www.techiedelight.com/find-minimum-cost-reach-last-cell-matrix-first-cell/)

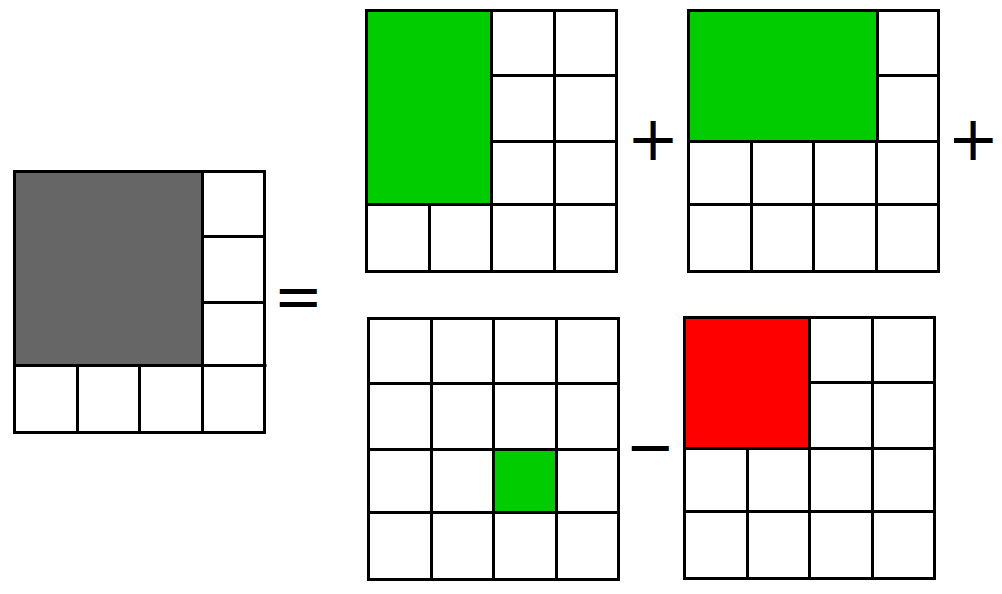
Построчно заполняем, учитывая, что могли прийти только слева или сверху

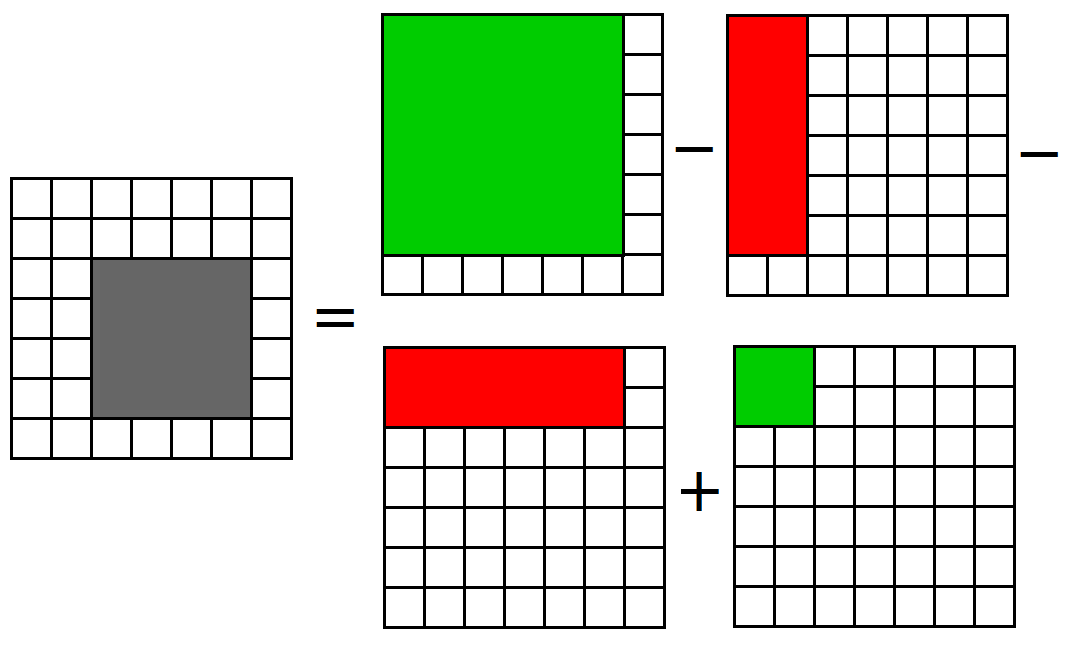
1. Дана матрица MxN. Для координат (p, q) и (r, s), обозначающих верхний левый и правый нижний угол подматрицы, необходимо вернуть сумму элементов этой подматрицы.

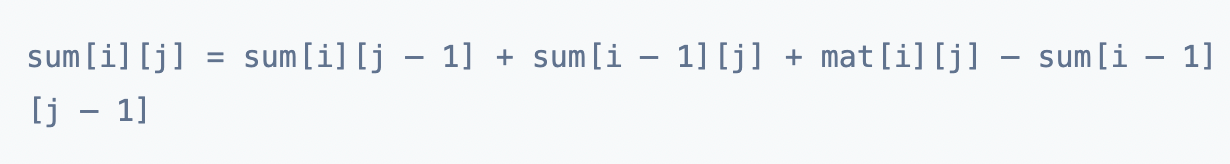


**Решение:** <https://www.techiedelight.com/calculate-sum-elements-sub-matrix-constant-time/>

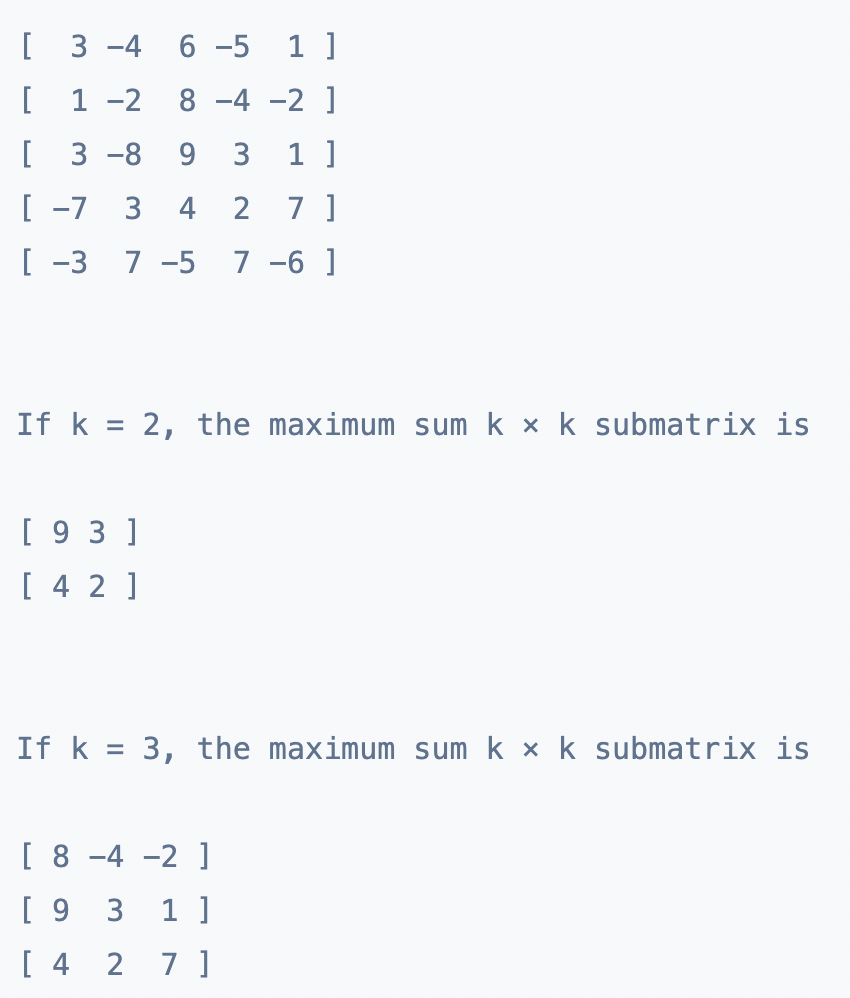
Идея: считаем в каждой ячейке матрицы сумму всех элементов матрицы с углами (0, 0) и текущая ячейка. При заполнении помогает арифметика, изображенная на двух картинках. Потом на все запросы отвечаем за константу, главное правильно формулы прописать





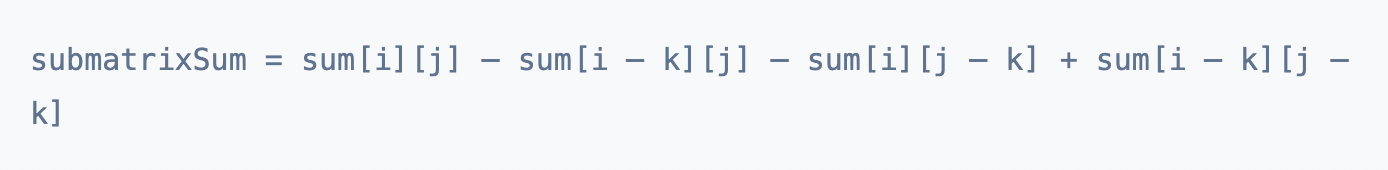


1. Дана матрица MxN. Необходимо вернуть максимальную сумму для подматрицы размера KxK.



**Решение**: <https://www.techiedelight.com/find-maximum-sum-submatrix-in-given-matrix/>

Ну зная предыдущую задачу все просто =)



1. Дана матрица MxN, заполненная нулями и единицами. Найти размер наибольшей квадратной матрицы, состоящей только из единиц. Для примера ниже ответ 3.

**Table

Description automatically generated**

**Решение:**

[**https://www.techiedelight.com/find-size-largest-square-sub-matrix-1s-present-given-binary-matrix/**](https://www.techiedelight.com/find-size-largest-square-sub-matrix-1s-present-given-binary-matrix/)