Задача о наибольшей общей подпоследовательности

Определение:

Последовательность $Z = \langle z_1, z_2, \ldots, z_k \rangle$ является **подпоследовательностью** (англ. *subsequence*) последовательности $X = \langle x_1, x_2, \ldots, x_m \rangle$, если существует строго возрастающая последовательность $\langle i_1, i_2, \ldots, i_k \rangle$ индексов X таких, что для всех $j = 1, 2, \ldots, k$ выполняется соотношение $x_{i_j} = z_j$.

Другими словами, подпоследовательность данной последовательности — это последовательность, из которой удалили ноль или больше элементов. Например, $Z = \langle B, C, D, B \rangle$ является подпоследовательностью последовательности $X = \langle A, B, C, B, D, A, B \rangle$, а соответствующая последовательность индексов имеет вид $\langle 2, 3, 5, 7 \rangle$.

Определение:

Последовательность Z является **общей подпоследовательностью** (англ. *common subsequence*) последовательностей X и Y, если Z является подпоследовательностью как X, так и Y.

Задача:

Пусть имеются последовательности $X = \langle x_1, x_2, \ldots, x_m \rangle$ и $Y = \langle y_1, y_2, \ldots, y_n \rangle$. Необходимо найти $LCS(X, Y)$

Теорема:

Пусть имеются последовательности $X = \langle x_1, x_2, \ldots, x_m \rangle$ и $Y = \langle y_1, y_2, \ldots, y_n \rangle$, а $Z = \langle z_1, z_2, \ldots, z_k \rangle$ — их LCS.

1. Если $x_m = y_n$, то $z_k = x_m = y_n$ и $Z_{k-1} - LCS(X_{m-1}, Y_{n-1})$
2. Если $x_m \neq y_n$, то из $z_k \neq x_m$ следует, что $Z - LCS(X_{m-1}, Y)$
3. Если $x_m \neq y_n$, то из $z_k \neq y_n$ следует, что $Z - LCS(X, Y_{n-1})$

Доказательство:

▷

1. Если бы выполнялось $z_k \neq x_m$, то к Z можно было бы добавить элемент $x_m = y_n$, и тогда получилась бы общая подпоследовательность длины $k + 1$, что противоречит предположению, что $Z - LCS$. Значит, выполняется $z_k = x_m = y_n$. Значит, Z_{k-1} — общая подпоследовательность X_{m-1} и Y_{n-1}. Докажем от противного, что $Z_{k-1} - LCS$: тогда есть общая подпоследовательность W, длина которой больше $k - 1$. Добавив к W элемент $x_m = y_n$, получим $LCS(X, Y)$, длина которой больше k (т.е. больше длины Z), что приводит к противоречию.
2. Если $z_k \neq x_m$, то Z — общая подпоследовательность X_{m-1} и Y. Пусть существует их общая подпоследовательность W, длина которой превышает k. Она также является общей подпоследовательностью X и Y, что противоречит предположению о том, что Z (длины k) — $LCS(X, Y)$.
3. Аналогично второму случаю.

◁

Решение [править]

Обозначим как $lcs[i][j]$ LCS префиксов данных последовательностей, заканчивающихся в элементах с номерами i и j соответственно. Получается следующее рекуррентное соотношение:

$$lcs[i][j] = \begin{cases} 0, & i = 0 \text{ or } j = 0 \\ lcs[i-1][j-1] + 1, & x[i] = y[j] \\ \max(lcs[i][j-1],\ lcs[i-1][j]), & x[i] \neq y[j] \end{cases}$$

Очевидно, что сложность алгоритма составит $O(mn)$, где m и n — длины последовательностей.

Построение подпоследовательности [править]

Для каждой пары элементов помимо длины LCS соответствующих префиксов хранятся и номера последних элементов, участвующих в этой LCS.Таким образом, посчитав ответ, можно восстановить всю наибольшую общую подпоследовательность.

Задача о наибольшей возрастающей подпоследовательности

Задача:

Дан массив из n чисел: $a[0..n-1]$. Требуется найти в этой последовательности строго возрастающую подпоследовательность наибольшей длины.

Определение:

Наибольшая возрастающая подпоследовательность (НВП) (англ. *Longest increasing subsequence, LIS*) строки x длины n — это последовательность $x[i_1] < x[i_2] < \cdots < x[i_k]$ символов строки x таких, что $i_1 < i_2 < \cdots < i_k, 1 \leqslant i_j \leqslant n$, причем k — наибольшее из возможных.

Решение за время O(N^2) [править]

Построим массив d, где $d[i]$ — это длина наибольшей возрастающей подпоследовательности, оканчивающейся в элементе, с индексом i. Массив будем заполнять постепенно — сначала $d[0]$, потом $d[1]$ и т.д. Ответом на нашу задачу будет максимум из всех элементов массива d. Заполнение массива будет следующим: если $d[i] = 1$, то искомая последовательность состоит только из числа $a[i]$. Если $d[i] > 1$, то перед числом $a[i]$ в подпоследовательности стоит какое-то другое число. Переберем его: это может быть любой элемент $a[j] (j = 0...i-1)$, но такой, что $a[j] < a[i]$. Пусть на каком-то шаге нам надо посчитать очередное $d[i]$. Все элементы массива d до него уже посчитаны. Значит наше $d[i]$ мы можем посчитать следующим образом: $d[i] = 1 + \max\limits_{j=0..i-1} d[j]$ при условии, что $a[j] < a[i]$.

Пока что мы нашли лишь максимальную длину наибольшей возрастающей подпоследовательности, но саму ее мы вывести не можем. Для восстановления ответа заведем массив $prev[0...n-1]$, где $prev[i]$ будет означать индекс в массиве $a[]$, при котором достигалось наибольшее значение $d[i]$. Для вывода ответа будем идти от элемента с максимальным значениям $d[i]$ по его предкам.

Решение за O(N log N) [править]

Для более быстрого решения данной задачи построим следующую динамику: пусть $d[i] (i = 0...n)$ — число, на которое оканчивается возрастающая последовательность длины i, а если таких чисел несколько — то наименьшее из них. Изначально мы предполагаем, что $d[0] = -\infty$, а все остальные элементы $d[i] = \infty$. Заметим два важных свойства этой динамики: $d[i-1] \leqslant d[i]$, для всех $i = 1...n$ и каждый элемент $a[i]$ обновляет максимум один элемент $d[j]$. Это означает, что при обработке очередного $a[i]$, мы можем за $O(\log n)$ с помощью двоичного поиска в массиве d найти первое число, которое больше либо равно текущего $a[i]$ и обновить его.

Для восстановления ответа будем поддерживать заполнение двух массивов: pos и $prev$. В $pos[i]$ будем хранить индекс элемента, на который заканчивается оптимальная подпоследовательность длины i, а в $prev[i]$ — позицию предыдущего элемента для $a[i]$.

Задача о рюкзаке

Задача:

Задача о рюкзаке *(англ. Knapsack problem) — дано N предметов, n_i предмет имеет массу $w_i > 0$ и стоимость $p_i > 0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.*

Формулировка задачи [править]

Дано N предметов, W — вместимость рюкзака, $w = \{w_1, w_2, \dots, w_N\}$ — соответствующий ему набор положительных целых весов, $p = \{p_1, p_2, \dots, p_N\}$ — соответствующий ему набор положительных целых стоимостей. Нужно найти набор бинарных величин $B = \{b_1, b_2, \dots, b_N\}$, где $b_i = 1$, если предмет n_i включен в набор, $b_i = 0$, если предмет n_i не включен, и такой что:

1. $b_1 w_1 + \dots + b_N w_N \leqslant W$
2. $b_1 p_1 + \dots + b_N p_N$ максимальна.

Метод динамического программирования [править]

Пусть $A(k, s)$ есть максимальная стоимость предметов, которые можно уложить в рюкзак вместимости s, если можно использовать только первые k предметов, то есть $\{n_1, n_2, \dots, n_k\}$, назовем этот набор допустимых предметов для $A(k, s)$.

$A(k, 0) = 0$

$A(0, s) = 0$

Найдем $A(k, s)$. Возможны 2 варианта:

1. Если предмет k не попал в рюкзак. Тогда $A(k, s)$ равно максимальной стоимости рюкзака с такой же вместимостью и набором допустимых предметов $\{n_1, n_2, \dots, n_{k-1}\}$, то есть $A(k, s) = A(k-1, s)$
2. Если k попал в рюкзак. Тогда $A(k, s)$ равно максимальной стоимости рюкзака, где вес s уменьшаем на вес k-ого предмета и набор допустимых предметов $\{n_1, n_2, \dots, n_{k-1}\}$ плюс стоимость k, то есть $A(k-1, s - w_k) + p_k$

$$A(k, s) = \begin{cases} A(k-1, s), & b_k = 0 \\ A(k-1, s - w_k) + p_k, & b_k = 1 \end{cases}$$

То есть: $A(k, s) = \max(A(k-1, s), A(k-1, s - w_k) + p_k)$

Стоимость искомого набора равна $A(N, W)$, так как нужно найти максимальную стоимость рюкзака, где все предметы допустимы и вместимость рюкзака W.

Восстановим набор предметов, входящих в рюкзак

Будем определять, входит ли n_i предмет в искомый набор. Начинаем с элемента $A(i, w)$, где $i = N$, $w = W$. Для этого сравниваем $A(i, w)$ со следующими значениями:

1. Максимальная стоимость рюкзака с такой же вместимостью и набором допустимых предметов $\{n_1, n_2, \dots, n_{i-1}\}$, то есть $A(i-1, w)$
2. Максимальная стоимость рюкзака с вместимостью на w_i меньше и набором допустимых предметов $\{n_1, n_2, \dots, n_{i-1}\}$ плюс стоимость p_i, то есть $A(i-1, w - w_i) + p_i$

Заметим, что при построении A мы выбирали максимум из этих значений и записывали в $A(i, w)$. Тогда будем сравнивать $A(i, w)$ с $A(i-1, w)$, если равны, тогда n_i не входит в искомый набор, иначе входит.

Метод динамического программирование всё равно не позволяет решать задачу за полиномиальное время, потому что его сложность зависит от максимального веса. Задача о ранце (или задача о рюкзаке) — одна из NP-полных задач комбинаторной оптимизации.

Пример [править]

$W = 13, N = 5$

$w_1 = 3, p_1 = 1$

$w_2 = 4, p_2 = 6$

$w_3 = 5, p_3 = 4$

$w_4 = 8, p_4 = 7$

$w_5 = 9, p_5 = 6$

	1	2	3	4	5	6	7	8	9	10	11	12	13
k = 0	0	0	0	0	0	0	0	0	0	0	0	0	0
k = 1	0	0	1	1	1	1	1	1	1	1	1	1	1
k = 2	0	0	1	6	6	6	7	7	7	7	7	7	7
k = 3	0	0	1	6	6	6	7	7	10	10	10	11	11
k = 4	0	0	1	6	6	6	7	7	10	10	10	13	13
k = 5	0	0	1	6	6	6	7	7	10	10	10	13	13

Числа от 0 до 13 в первой строчке обозначают вместимость рюкзака.

В первой строке как только вместимость рюкзака $n \geqslant 3$, добавляем в рюкзак 1 предмет.

Рассмотрим $k = 3$, при каждом $s \geqslant 5$(так как $w_3 = 5$) сравниваем $A[k-1][s]$ и $A[k-1][s - w_3] + p_3$ и записываем в $A[k][s]$ стоимость либо рюкзака без третьего предмета, но с таким же весом, либо с третьим предметом, тогда стоимость равна стоимости третьего предмета плюс стоимость рюкзака с вместимостью на w_3 меньше.

Максимальная стоимость рюкзака находится в $A(5, 13)$.