# Алгоритмы и Структуры Данных ДЗ-9

Гарипов Роман M3138

24.11.2019

## Задача №1

Посчитаем $dp_i$ = кол-во различных подпоследовательностей на префиксе $[0 : i]$. $dp[0] = 1$, так как всего последовательностей на таком префиксе ровно 1. Рекурентная формула для этой динамики выглядит так :

$$dp[i-1] = \begin{cases} 2 * dp[i-1] + 1, \text{ если } a[i] \text{ не встречалось на префиксе } [0; i-1]; \\ 2 * dp[i-1] - dp[x-1], \text{ x = последнее вхождение } a[i], \text{ иначе;} \end{cases}$$

В случае если для очередного $i$, $a[i]$ не встречалось на префиксе, мы можем просто взять любую подпоследовательность на префиксе $[0; i-1]$ и добавить или не добавить к каждой из них $a[i]$, а так же прибавить подпоследовательность из одного элемента $a[i]$, поэтому формула в этом случае $dp[i] = 2 * dp[i-1] + 1$.
В случае когда $a[i]$ уже было на префиксе, мы поступим точно так же, возьмем все подпоследовательности на префиксе и добавим или не добавим к ним $a[i]$, $dp[i] = dp[i-1]*2$. Но тогда мы посчитали некоторые подпоследовательности два раза. Пусть $x$ так же как и в формуле - последнее вхождение $a[i]$. Тогда понятно, что мы посчитали два раза все подпоследовательности на префиксе $[0; x-1]$, с добавленным элементом $a[i]$ на конце, так как они были посчитаны при переходе из $dp[x-1]$ в $dp[x]$. Вычтем их и получим верную формулу. $dp[i] = 2 * dp[i-1] - dp[x-1]$.
Вычисление состояние динамики по этим формулам верно, потому что мы посчитаем каждую подпоследовательность ровно один раз, как раз для этого мы вычитаем те что посчитали дважды. А так же, никакую подпоследовательность мы не упустим, потому что прибавляем 1 в формуле для ещё не встреченного $a[i]$, добавляя все последовательности из одного элемента. Так посчитаются все различные подпоследовательности.
Таким образом, чтобы решить эту задачу, необходимо на каждой итерации цикла посчитать состояние динамики и обновить последнее вхождение последнего элемента на рассмотренном префиксе.
Ответ будет равен $dp[n-1]$.

## Задача №2

## Задача №3

Будем делать примерно как в НВП за $\mathcal{O}(n \log(n))$ с бинпоиском, но немного по-другому. Вместо того чтобы уменьшать для каждой длины элемент на который заканчивается ВП этой длины, будем хранить все элементы, которые стояли в нашей динамике в этой позиции.
Чтобы делать бинпоиск, будем использовать элемент на конце вектора, чтобы наш алгоритм работал корректно, ведь в обычном алгоритме мы просто заменяли на меньший элемент, а это и будет минимальный элемент в этом

векторе, так как мы добавляем значение вектор так, что они упорядочены по убыванию(потому что находим в динамике минимальный элемент больший либо равный нашего). Поскольку изначалально каждый вектор состоит из одного элемента $inf$, для удобства будем удалять его при добавлении первого значения в соответсвующий вектор.

Теперь мы храним вектора вместо одного значения. Для каждого элемента запомним под каким номером он мог быть в НВП.

Построим примерно таким же алгоритмом убывающую последовательность, но будем идти с конца. И запомним для каждого элемента под каким номером он может встретиться в этой убывающей последовательности. Для каждой длины НВП будем запоминать сколько элементов могут стоять на этой позиции. Пройдемся по всем элементам векторов которые построили для первой последовательности, если номер элемента в возрастающей последовательности + номер элемента в убывающей последовательности = длине НВП - 1, тогда этот элемент точно встречается в НВП, запишем что он может стоять на своей длине в НВП.

Пройдемся по длинам, если для текущей длины есть больше 1 элемента претендующего встать на конец, понятно что все элементы которые могут встать на эту позицию встречаются не во всех НВП, для них запишем что они встрчаются в каких-то НВП. Если ровно 1, то такой элемент встречается во всех НВП, для него это и запишем. Остальные элементы в НВП не входят.