Создание команды

Переформулируем более формально условие задачи. Требуется выбрать подпоследовательность массива, где для каждого числа, кроме первого, должно существовать число среди предыдущих, не меньше текущего. Поймем, что это тоже самое, что первое число в подпоследовательности является максимальным. Действительно, для второго числа существует только одно число, которое должно быть больше или равно его, а именно первое. Для третьего числа подходят первое или второе, но первое всегда больше или равно второго, поэтому нужно, чтобы первое было больше или равно третьему. Для произвольного размера подпоследовательности это рассуждение можно произвести по индукции.

Нужно теперь научиться искать подпоследовательность с этим свойством максимальной длины. Одним проходом по массиву можем построить подпоследовательность максимальной длины, у которой первое число совпадает с первым числом из массива. Будем перебирать первое число в подпоследовательности слева направо. Пусть у предыдущей последовательности первый индекс $last$, а текущий индекс tmp, тогда есть смысл строить новую подпоследовательность, если $a[tmp] > a[last]$, так как в ином случае tmp принадлежит подпоследовательности с элементом $last$ и не сможем увеличить ответ. Тогда произойдут два изменения в подпоследовательности : пропадут все элементы с индексами из $[last + 1; tmp - 1]$ и добавится часть элементов с отрезка $[tmp + 1; n]$.

Элементы можем убрать, перебрав индексы из отрезка и вычтя 1 из текущей длины, если $a[i] < a[last]$, так как в таком случае они принадлежали текущей подпоследовательности и они больше не войдут. Суммарно затраченное время на это будет $O(n)$, так как каждый раз $last$ увеличивается до tmp.

Осталось добавить новые элементы. Для этого переберём все возможные элементы, которые могут войти циклом от $a[last]+1$ до $a[tmp]$, переберём все индексы каждого элемента и, если $i > last$, то можем увеличить подпоследовательность. Суммарно затраченное время на это будет $O(n)$, так как каждый раз $last$ увеличивается до tmp и все числа в массиве от 1 до n.

Значит, общее время работы алгоритма будет $O(n)$, так как делаем перебор длины $O(n)$ суммарно работающий за $O(n)$.