Рогуленко Виктор
+7 929 670 17 04

Алгоритмы и структуры данных 2018

Домашняя работа №2. Задача B.

Условие Вы хотите набрать футбольную команду. У каждого игрока своя эффективность, она описывается одним целым числом. Чем больше число, тем больше эффективность футболиста. Обязательным условием для любой команды является сплоченность. Если один из игроков играет сильно лучше всех остальных, его будут недолюбливать, и команда распадется. Поэтому эффективность любого игрока команды не должна превышать сумму эффективностей любых двух других игроков (условие $*$). Ваша задача — набрать команду, которая будет удовлетворять условию сплоченности, и при этом иметь наибольшую суммарную эффективность.

Формат ввода. В первой строке входа задано число n ($1 \leq n \leq 100000$). Во второй строке — эффективности каждого из n игроков — положительные целые числа, не превосходящие $2^{31}-1$.

Формат вывода. Выведите две строки. В первую запишите наибольшую возможную сумму эффективностей игроков в команде. Во вторую строку выведите номера всех игроков, которых нужно взять в команду, в порядке возрастания.

Решение

Отсортируем игроков по неубыванию эффективности, запомнив их исходные индексы в отдельном массиве $LegacyIndeces$ для последующего вывода. Докажем, следующее

Утверждение. Одним из решений (возможно, единственным) всегда будет некая подпоследовательность из подряд идущих элементов (игроков).

Пусть $a_1 \leq a_2 \leq ... \leq a_n$ - отсортированный массив эффективностей, а $a_{i_1} \leq a_{i_2} \leq ... \leq a_{i_k}$ - какая-то подпоследовательность, удовлетворяющая условию $(*)$ и имеющая наибольшую возможную суммарную эффективность $S_i = S_{max}$. Из условия $(*)$ следует, что

$$\forall j = 1, 2, ..., k : a_{i_j} \leq (a_{i_1} + a_{i_2}).$$

Первые два элемента подпоследовательности как самые малые ее величины диктуют ограничение на всю подпоследовательность. Тогда все элементы отсортированного массива a от a_{i_2} до a_{i_k} входят в a_i, ведь в противном случае недостающий элемент можно было бы добавить без нарушения условия $(*)$ и увеличить S_i сверх S_{max}.

Заметим также, что если $i_1 + 1 < i_2$, то заменим i_1 на $i_2 - 1$. Полученная подпоследовательность будет также удовлетворять условию $(*)$, поскольку мы ослабили ограничение первых двух элементов. Кроме того, ее сумма будет не меньше S_{max} в силу неубывания a. Утверждение доказано. □

Задача свелась к нахождению подпоследовательности упорядоченного по неубыванию массива a, удовлетворяющей условию $(*)$ и имеющей наибольшую возможную сумму. Для этого сделаем предподсчет массива частичных сумм $PartialSum_i = \sum_{j=1}^{i} a_j$, $PartialSum_0 = 0$ и инициализируем $S_{max} = a_1 + a_2$, $(i_{max}, j_{max}) = (1, 2)$. Далее переберем элементы a_i от a_1 до a_{n-1} и бинарным поиском на массиве от a_{i+3} до a_n найдем максимальный элемент a_j такой, что $a_j \leq (a_i + a_{i+1})$. С помощью массива частичных сумм находим $S_i = PartialSum_j - PartialSum_{i-1}$. Если $S_i > S_{max}$, обновляем S_{max} и запоминаем пару $(i_{max}, j_{max}) = (i, j)$. По окончании перебора выводим S_{max} и все элементы $LegacyIndeces[i]$ от i_{max} до j_{max}, что и является искомым решением.

Сложность по времени: $O(n \log n)$. При $(1 \leq n \leq 100000)$ время работы алгоритма уложится в отведенные ограничения. Рассмотрим, из чего складывается оценка по времени.

Отсортируем исходный массив алгоритмом быстрой сортировки за $O(n \log n)$. Предподсчет массива частичных сумм выполним за $O(n)$. Поиск искомой подпоследовательности выполним за $O(n \log n)$, перебрав n индексов и произведя бинарный поиск не более чем за $O(\log n)$. Алгоритм можно ускорить, если запускать бинарный поиск не на (a_{i+3}, a_n), а на (a_j, a_n) (каждая следующая рассматриваемая подпоследовательность заканчивается не раньше предыдущей).

Сложность по памяти: $O(n)$. Потребуются следующие массивы длины n:

- исходный массив (unsigned int, поскольку все элементы целые и положительные),
- отсортированный массив a (unsigned int; при необходимости можно выполнить сортировку in-place),
- массив исходных индексов $LegacyIndeces$ (unsigned int),
- массив частичных сумм $PartialSum$ (unsigned long int, чтобы избежать возможного переполнения).