H1, разбор контеста

simagin.mail@yandex.ru

A02.25

A1

В задаче речь идет о блинной сортировке. Самый простейший ее вариант имеет квадратичное решение и напоминает сортировку вставками.

Изначально массив не отсортирован. Мы выбираем наибольший элемент и с помощью двух переворотов ставим его в конец. Для этого достаточно поместить максимальный элемент в начало массива первым переворотом, а затем развернуть весь массив. Теперь осталось отсортировать $n - 1$ элемент. Повторяя подобные действия, упорядочим весь массив.

Очевидно, что нам потребуется не более $2n$ переворотов. Количество действий на каждом этапе можно оценить $O(n)$. А значит сложность всей сортирвоки $O(n^2)$.

A2

Упорядочим n наклеек Диего по убыванию за $O(n \log n)$. На семинаре мы разбирали, как можно удалить не уникальные элемента за $O(n)$, проделаем это. А после упорядочим k коллекционеров по убыванию минимального номера наклейки за $O(k \log k)$.

Установим указатель на начало массива с наклейками Диего. И будем двигать указатель до тех пор, пока не найдем наклейку, которая обладает меньшим номером, чем наклейка первого коллекционера. Очевидно, что наклеек. начиная с наклейки под указателем, у коллекционера нет. Найдем количество наклеек, которое мог бы взять первый коллекционер у Диего. Так как мы предусмотрительно упорядочили коллекционеров, то продолжая двигать указатель дальше добьемся того же для второго коллекционера и т.д. Таким образом вы найдем решение задачи за $O(\max(k, n))$.

Совокупную сложность алгоритма можно оценить, как $O(m \log m)$, где $m = \max(k, n)$.

Замечение. Кстати, хранить и сортировать коллекционеров не обязательно. Достаточно модернизировать алгоритм бинарного поиска для вычисления количества марок, которое можно взять у Диего. Однако предложенное выше решение позволяет свести асимптотичскую сложность алгоритма к $O(m)$, если в качестве сортировки использовать Radix Sort. Но на практике это не имеет смысла.

B1

Задача является интерпретацией задачи о покрытии прямой отрезками. Ее решение разбиралось на семинаре. Сложность алгоритма $O(n \log n)$, однако может быть сведена к $O(n)$, если для упорядочивания точек (досок) использовать сортировку с линейной сложностью.

B2

Упорядочим всех певцов по возрастанию веса. И инициализируем решение. Будем идити по массиву весов, пока не найдем для каждого города хотя бы по одному исполнителю. Для каждого города возьмем певца, который встретился последним. Это и будет начальное решение.

Теперь будем двигаться по массиву дальше и пытаться улучшить решение. Для этого мы будем запоминать для каждого города, последний вес исполнителя, который нам встретился. Если разница максимального и минимального веса в такой четверки меньше, чем у решения, то обновим его. Обработав всех певцов таким образом, получим решение задачи.

Очевидно, что сложность алгоритма $O(n \log n)$. Что же делает алгоритм? Он фиксирует позицию в упорядоченном массиве певцов, а затем находит минимальный по длине массив, который заканчивается на рассматриваемом элементе, содержит хотя бы по одному певцу из каждого города и имеет минимальную длину. Очевидно, что действуя таким образом, мы получим истинное решение. Для доказательство этого можно использовать метод от противного.