Оглавление

Размер массива

Дан массив A длины n. Найдите количество его различных элементов.

Задача решается тривиально: заносим все элементы в множество (set) или используем словарь (map) для подсчёта уникальных значений. Количество ключей в словаре и будет ответом.

Количество элементов всех типов

Дан массив A длины n. Для каждого элемента найдите, сколько раз он встречается в массиве.

Используем словарь (map) для подсчёта частот: проходим по массиву и увеличиваем счётчик для каждого значения. Затем выводим все пары (значение, частота) в порядке возрастания значений. Для этого можно отсортировать уникальные значения или просто пройти по отсортированному массиву, выводя каждый элемент только один раз.

Прибавь слева прибавь справа

Изначально пустой массив. Запросы: добавить элемент в начало или в конец. Вывести итоговый массив.

Разделим запросы на два типа: добавляемые слева и справа. Для добавлений слева порядок будет обратным, так как каждый новый элемент становится перед предыдущими. Поэтому собираем все левые добавления в отдельный вектор, затем разворачиваем его. Правые добавления просто добавляем в конец в порядке поступления. Затем выводим сначала левые (в обратном порядке), потом правые.

Спой, птичка!

Оценки птиц различны. Процесс: слушаем птиц по очереди. Если текущая оценка меньше текущего максимума, то после неё сразу же повторно слушаем птицу с максимальной оценкой. Найти общее количество прослушиваний и максимальное число прослушиваний одной птицы.

Идём по массиву оценок в порядке первого прослушивания. Отслеживаем текущий максимум mx и сколько раз каждая птица была прослушана (в словаре m). Если новая оценка больше mx, то это новое первое прослушивание, просто добавляем 1 к счётчику и обновляем максимум. Если новая оценка меньше mx, то после неё мы ещё раз слушаем птицу с максимумом, поэтому добавляем 2 к общему числу прослушиваний и увеличиваем счётчик для текущей максимальной птицы. В конце выводим общее число и максимум среди m.

Кирилл и кролики

Каждый кролик за один приём съедает одно и то же целое число морковок от a до b. За два последних приёма суммарно съедено n морковок. Найти максимально возможное количество кроликов или -1, если это невозможно.

Пусть x — количество морковок на одного кролика, k — количество кроликов. Тогда за два приёма съедено $2kx = n$, значит $x = \frac{n}{2k}$. x должно быть целым и лежать в $[a, b]$, поэтому $a \le \frac{n}{2k} \le b$.

Преобразуем неравенства:

- $\frac{n}{2k} \ge a \Rightarrow k \le \frac{n}{2a}$
- $\frac{n}{2k} \le b \Rightarrow k \ge \frac{n}{2b}$

Также n должно быть чётным, иначе ответ -1. Ищем максимальное целое k, удовлетворяющее $\frac{n}{2b} \le k \le \frac{n}{2a}$. Если таких k нет, то -1, иначе ответ $\lfloor \frac{n}{2a} \rfloor$.

Лягушка и ягоды

Лягушка прыгает с 1-й по n-ю кочку. Обычно на следующую, но один раз может прыгнуть на k вперёд. На каждой кочке получает a_i. Найти максимальную сумму.

Посчитаем префиксные суммы $pref[i] = a_1 + a_2 + ... + a_i$.

Если лягушка делает длинный прыжок с кочки i на кочку $i + k$, то она пропускает кочки с $i + 1$ по $i+k-1$. Путь будет: кочки $1, 2, ..., i$, затем прыжок на $i+k$, затем $i+k+1, ..., n$. Сумма: $pref[i]+(pref[n]-pref[i + k - 1])$.

Перебираем все возможные i от 1 до $n - k$ (чтобы $i + k \leq n$). Также не забываем вариант без прыжка — просто $pref[n]$. Берём максимум из всех вариантов.

МегаПокер

В ряд лежат n карточек с числами от 1 до m. Можно удалять карточки, не меняя порядок оставшихся. Нужно, чтобы оставшиеся образовали последовательность подряд идущих чисел (каждое на 1 больше предыдущего). Найти максимальную длину такой последовательности.

Задача сводится к поиску самой длинной подпоследовательности (не подряд, но с сохранением порядка), которая является последовательными числами. Для каждого числа x храним длину максимальной последовательности, заканчивающейся на x. Тогда при просмотре карточки x обновляем $dp[x] = dp[x-1]+1$. В конце берём максимум по всем $dp[x]$.

Замени на сумму

Даны две последовательности. За одну секунду можно в любой из них заменить два соседних элемента на их сумму (длина уменьшается на 1). Можно ли не более чем за k секунд сделать последовательности равными?

Сначала проверим, что суммы элементов равны, иначе сразу NO.

Процесс слияния соседних элементов эквивалентен разбиению исходных последовательностей на одинаковые по сумме блоки. Нужно, чтобы можно было разбить обе последовательности на одинаковое количество блоков с равными суммами. Количество операций слияния равно $(n - t) + (m - t)$, где t — количество блоков (они же — количество элементов в итоговых равных последовательностях). Чем больше t, тем меньше операций.

Жадно идём по обеим последовательностям, накапливая текущие суммы. Когда суммы сравниваются, начинаем новый блок. Если суммы не равны, добавляем следующий элемент из той последовательности, где сумма меньше. Если в какой-то момент не хватает элементов, чтобы уравнять суммы — ответ NO.

После того как разбили на t блоков, считаем минимальное количество операций: $(n-t)+(m-t)$. Если оно $\leq k$, то YES, иначе NO.

Монстры

У каждого монстра три параметра: a_i, b_i, c_i. Если у игрока k монстров, то для монстра i его агрессивность равна a_i при $k \leq c_i$ и b_i при $k > c_i$. Нужно выбрать k монстров с суммарной агрессивностью $\leq s$. Найти максимальное k.

Бинарный поиск по k - для фиксированного k вычисляем для каждого монстра его стоимость: $cost_i = a_i$, если $k \leq c_i$, иначе b_i. Затем выбираем k монстров с минимальными стоимостями и проверяем, не превышает ли сумма s. Если да, то k достижимо. Сложность: $O(n \log n)$ на проверку, всего $O(n \log n \log n)$.

Пиратские сундуки

Есть n жетонов и m сундуков в ряд. Открыть очередной сундук стоит 1 жетон. Можно пропустить $N-1$ сундуков и открыть N-й, заплатив $1+2+...+N$ жетонов. При открытии i-го сундука получаем a_i очков. Найти максимальную сумму очков.

Это задача на рюкзак: состояние $dp[i][j]$ — максимальная сумма очков, которую можно получить, открыв какие-то сундуки среди первых i, потратив ровно j жетонов. Переход: если мы хотим следующим открыть сундук $i+k$, то нужно заплатить $cst = k(k+1)/2$ жетонов за пропуск $k-1$ сундуков и открытие k-го. Добавляем a_{i+k} к сумме. В конце берём максимум по всем i и j.

Медведь и правильное питание

Есть n продуктов. i-й продукт имеет время исчезновения t_i и калорийность k_i. Можно съедать не более одного продукта в час. Продукт доступен, если текущее время $\leq t_i$. Найти максимальную суммарную калорийность.

Сортируем продукты по времени исчезновения. Проходим по ним и добавляем текущую калорийность в мультимножество. Если размер множества превышает текущее t_i (то есть мы пытаемся съесть больше продуктов, чем доступно часов), удаляем самый маленький по калорийности элемент (жадно выгоднее оставить более калорийные). В конце суммируем все оставшиеся в множестве калорийности — это и будет ответ.