**Задачи по сортировки на собеседованиях**

**Панчиков Иван**

1. **Сортировка массива с элементами 0, 1 и 2 (Задача Датч-флага)**

Задача Датч-флага (Dutch National Flag problem) — это классическая задача сортировки трёх типов элементов.

Отсортируйте массив, содержащий только 0, 1 и 2, без использования встроенной сортировки.

Решение:

def sortColors(nums):  
 low, mid, high = 0, 0, len(nums) - 1  
 while mid <= high:  
 if nums[mid] == 0:  
 nums[low], nums[mid] = nums[mid], nums[low]  
 low += 1  
 mid += 1  
 elif nums[mid] == 1:  
 mid += 1  
 else:  
 nums[mid], nums[high] = nums[high], nums[mid]  
 high -= 1

Частые ошибки:

* Ошибка с условием окончания цикла: нужно “mid <= high”, а не “mid < high”.
* Ошибка при обмене: забывают обновлять индексы “low”, “mid”.

1. **Сортировка почти отсортированного массива (каждый элемент не дальше k от своей позиции)**

**Min-heap** (минимальная куча) — это **специальная структура данных**, обычно реализованная через бинарную кучу, где:

* Каждый родитель ≤ своих детей
* Значение в **корне (на вершине)** — **наименьшее** во всей куче

Используйте min-heap (кучу) для сортировки массива, где каждый элемент не дальше, чем на k позиций от своей отсортированной позиции.

**Решение:**

import heapq  
def sort\_k\_sorted\_array(arr, k):  
 heap = arr[:k+1]  
 heapq.heapify(heap)  
 target\_idx = 0  
  
 for i in range(k+1, len(arr)):  
 arr[target\_idx] = heapq.heappop(heap)  
 heapq.heappush(heap, arr[i])  
 target\_idx += 1  
  
 while heap:  
 arr[target\_idx] = heapq.heappop(heap)  
 target\_idx += 1

Частые ошибки:

* Забывают, что начальная куча должна быть “arr[:k+1]”, а не “arr[:k]”.
* Ошибка с “target\_idx” — неправильный счётчик записи.

**3. Сортировка строк по custom алфавиту**

Дан алфавит в виде строки. Отсортируйте массив строк по этому алфавиту.

**Решение:**

def custom\_sort(strings, order):  
 order\_map = {ch: i for i, ch in enumerate(order)}  
 def transform(s):  
 return [order\_map[c] for c in s]  
 return sorted(strings, key=transform)

Частые ошибки:

* Ошибка при маппинге символов — не проверяют, что все символы есть в “order”.
* Путают “sorted()” с “list.sort()” (in-place vs return).

Можно использовать метод **list.sort()**

**Это метод списка (только у объектов типа list), который:**

* Изменяет список на месте.
* Не возвращает ничего (None).

nums = [3, 1, 2]  
nums.sort()  
print(nums) # [1, 2, 3]

**4. Поиск k-го наименьшего элемента (QuickSelect)**

**QuickSelect** — это алгоритм для поиска **k-го по порядку элемента** (например, k-го наименьшего) **без полной сортировки** массива. Он основан на том же принципе, что и QuickSort.

Найдите k-й наименьший элемент в неотсортированном массиве за среднее время O(n).

**Решение:**

import random  
def quickselect(arr, k):  
 if len(arr) == 1:  
 return arr[0]  
 pivot = random.choice(arr)  
 lows = [el for el in arr if el < pivot]  
 highs = [el for el in arr if el > pivot]  
 pivots = [el for el in arr if el == pivot]  
  
 if k < len(lows):  
 return quickselect(lows, k)  
 elif k < len(lows) + len(pivots):  
 return pivots[0]  
 else:  
 return quickselect(highs, k - len(lows) - len(pivots))

Частые ошибки:

* Не учитывают повторы (pivot может встречаться несколько раз)
* Ошибки в смещении индекса `k` при рекурсии.