from math import sqrt

from typing import List, Iterable, TypeVar

T = TypeVar("T", int, float)

def insertion\_sort(a: List[T]) -> None:

"""

Стабильная сортировка вставками (in-place) для содержимого одной корзины.

В среднем O(k^2) на корзину, где k — размер корзины.

"""

for i in range(1, len(a)):

key = a[i]

j = i - 1

# Сдвигаем элементы вправо, пока не найдём позицию для key

while j >= 0 and a[j] > key:

a[j + 1] = a[j]

j -= 1

a[j + 1] = key

def bucket\_sort(arr: Iterable[T], bucket\_count: int | None = None) -> List[T]:

"""

Блочная (корзинная) сортировка.

Идея: распределяем элементы по корзинам по диапазону значений -> сортируем каждую корзину ->

склеиваем корзины слева направо.

:param arr: итерируемая последовательность чисел (int/float), допускаются отрицательные значения

:param bucket\_count: количество корзин; по умолчанию ~ sqrt(n) — сбалансированная эвристика

:return: новый отсортированный список

"""

a = list(arr) # не мутируем вход

n = len(a)

if n <= 1:

return a.copy()

# Эвристика: число корзин ~ sqrt(n) (можно переопределить параметром)

if bucket\_count is None:

bucket\_count = max(1, int(sqrt(n)))

# Находим диапазон значений

mn = min(a)

mx = max(a)

if mn == mx:

# Все элементы равны — уже отсортировано

return a.copy()

rng = mx - mn

# Создаём пустые корзины

buckets: List[List[T]] = [[] for \_ in range(bucket\_count)]

# Функция распределения: нормализуем x в [0, 1] и масштабируем в индексы [0 .. bucket\_count-1]

# Важно: крайние значения (особенно mx) должны попадать в последнюю корзину.

for x in a:

# При rng > 0 значение ((x - mn) / rng) в диапазоне [0, 1].

# Умножаем на (bucket\_count - 1), чтобы верхняя граница ушла в последнюю корзину.

idx = int((x - mn) / rng \* (bucket\_count - 1))

buckets[idx].append(x)

# Сортируем каждую корзину алгоритмом вставок (можно заменить на другой при желании)

for b in buckets:

insertion\_sort(b)

# Склеиваем корзины слева направо

result: List[T] = []

for b in buckets:

result.extend(b)

return result

if \_\_name\_\_ == "\_\_main\_\_":

# Демонстрация: разные наборы данных

tests = [

[0.42, 0.32, 0.23, 0.52, 0.25, 0.47, 0.51],

[3, 1, 4, 1, 5, 9, 2, 6],

[-3.5, 2.2, 0.0, -1.1, 4.8, 2.2, -3.6],

[10, 10, 10, 10],

]

for i, arr in enumerate(tests, 1):

print(f"Тест {i}: исходные данные = {arr}")

sorted\_arr = bucket\_sort(arr) # можно передать bucket\_count=... при необходимости

print(f"Тест {i}: отсортировано = {sorted\_arr}\n")

РЕЗУЛЬТАТ РАБОТЫ КОДА:

Тест 1: исходные данные = [0.42, 0.32, 0.23, 0.52, 0.25, 0.47, 0.51]

Тест 1: отсортировано = [0.23, 0.25, 0.32, 0.42, 0.47, 0.51, 0.52]

Тест 2: исходные данные = [3, 1, 4, 1, 5, 9, 2, 6]

Тест 2: отсортировано = [1, 1, 2, 3, 4, 5, 6, 9]

Тест 3: исходные данные = [-3.5, 2.2, 0.0, -1.1, 4.8, 2.2, -3.6]

Тест 3: отсортировано = [-3.6, -3.5, -1.1, 0.0, 2.2, 2.2, 4.8]

Тест 4: исходные данные = [10, 10, 10, 10]

Тест 4: отсортировано = [10, 10, 10, 10]