МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

| старший преподаватель |  |  |  | С.Ю. Гуков |
| --- | --- | --- | --- | --- |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| --- |
| Максимум в скользящем окне |
| по курсу: |
| АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ гр. № | 4329 |  |  |  | Д.С. Шаповалова |
| --- | --- | --- | --- | --- | --- |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

ОГЛАВЛЕНИЕ

[Цель работы 3](#_30j0zll)

[Постановка задачи 3](#_1fob9te)

[Схема алгоритма решения 4](#_3znysh7)

[Полное описание реализованной функции 4](#_gqc1nhfmq022)

[Листинг программы 5](#_2et92p0)

[Результат выполнения программы. 5](#_tyjcwt)

[ВЫВОДЫ 6](#_3dy6vkm)

# Цель работы

Написать программу, ищущую максимум в каждом окне размера m данного массива чисел A[1 ... n].

# Постановка задачи

Задание: Написать программу, ищущую максимум в каждом окне размера m данного массива чисел A[1 ... n]. Текст задания приведён в таблице 1.

Таблица 1. Индивидуальное задание

| № | Текст задания | Вход | Выход |
| --- | --- | --- | --- |
| 0 | Написать программу, ищущую максимум в каждом окне размера m данного массива чисел A[1 ... n].  Наивный способ решить данную задачу – честно просканировать каждое окно и найти в нём максимум. Время работы такого алгоритма – O(nm). **Ваша задача – реализовать алгоритм со временем работы O(n).**  *Ограничения.* 1 ≤ n ≤ 105, 1 ≤ m ≤ n, 0 ≤ A[i] ≤ 105 для всех 1 ≤ i ≤ n. | Первая строка входа содержит число n, вторая – массив A[1 ... n], третья – число m. | n−m+1 максимумов, разделённых пробелами. |

# Схема алгоритма решения

1. Чтение данных из консоли
2. Функция поиска максимума
3. Вывод массива максимумов

# Полное описание реализованной функции

**Импорт класса deque из модуля collections**: deque — это двусторонняя очередь, позволяющая эффективно добавлять и удалять элементы с обоих концов (сначала и с конца).

**Функция поиска максимума sliding\_window\_maximum:**

Принимает три параметра: n — количество элементов в массиве, A — сам массив чисел, m — размер окна.

**Инициализируем пустую двустороннюю очередь**- deque(), которая будет использоваться для хранения индексов элементов массива. Индексы упорядочены так, что элементы по этим индексам в массиве «A» всегда будут отсортированы в убывающем порядке.

**Проходим циклом** по элементам массива A, от 0 до n-1. *Это дает вклад O(n).*

**Удаление старых индексов:** проверяем, что dq не пуст, затем проверяем, вышел ли первый элемент в деке за пределы текущего окна. Если dq[0] (индекс наименьшего элемента) меньше, чем i - m + 1, то мы удаляем его с помощью dq.popleft(), так как этот индекс больше не относится к текущему окну. *Эта операция в среднем выполняется за O(1).*

**Удаление меньших элементов:** пока dq не пуст и текущий элемент больше элемента, находящегося по индексу dq[-1] (последний индекс в деке), мы удаляем его из дека, вызывая dq.pop(). Так ищем максимум. *Суммарная сложность всех удалений не превышает O(n).*

**Добавление текущего индекса:** После очищения дека мы добавляем текущий индекс i в дек, так как он может стать кандидатом на максимум в текущем окне. *Операция выполняется за O(1).*

**Добавление максимума в результаты:**проверяем, сформировалось ли полное окно. Если индекс i больше или равен m-1, значит, текущий индекс показывает, что окно полностью заполнилось. Значит добавляем в список результатов значение элемента в массиве A, которое соответствует индексу первого элемента дека (dq[0]), так как именно он является текущим максимумом окна.

**Возвращаем**список max\_numbers с максимумами для каждого окна.

*Общая временная сложность алгоритма равна O(n).*

# Листинг программы

from collections import deque

def sliding\_window\_maximum(n, A, m):

# Дек для хранения индексов элементов массива

dq = deque()

max\_numbers = []

for i in range(n):

# Удаляем элементы из дека, которые выходят за пределы текущего окна

if dq and dq[0] < i - m + 1:

dq.popleft()

# Удаляем элементы из дека, которые меньше текущего элемента

while dq and A[dq[-1]] < A[i]:

dq.pop()

# Добавляем текущий индекс в дек

dq.append(i)

# Если окно полностью заполнено, добавляем максимум в результат

if i >= m - 1:

max\_numbers.append(A[dq[0]])

return max\_numbers

# Чтение входных данных

n = int(input()) # Число элементов

A = list(map(int, input().split())) # Сам массив

m = int(input()) # Размер окна

# Получаем максимумы и выводим результат

result = sliding\_window\_maximum(n, A, m)

print(" ".join(map(str, result)))

# 

# Результат выполнения программы.

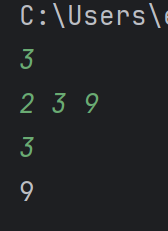


Рисунок 2.1 - 1-ый результат работы программы

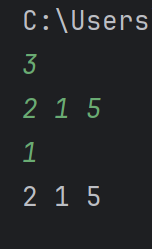
****

Рисунок 2.2 - 2-ой результат работы программы

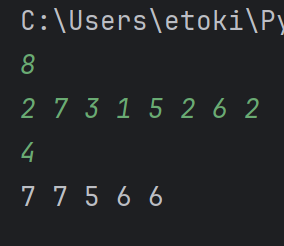


Рисунок 2.3 - 3-ой результат работы программы

# Выводы

В ходе выполнения лабораторной работы мной были освоены и изучены: класс deque; алгоритм нахождения максимума в скользящем окне. Написанная программа была протестирована, полученный результат соответствует значению в примере.