**Лабораторная №3**

**Выполнил: Ягодяк Павел Сергеевич АИСа-020303-о24**

**Использование различных структур данных для решения задачи замены элементов массива на ближайшие большие.**

**Цель:**

Изучить и реализовать структуры данных (массив, связанный список и стандартные библиотеки) для решения задачи, а также сравнить их производительность.

**Задание:**

В таблице A размера N за один просмотр необходимо каждый элемент заменить на ближайший следующий за ним элемент, который больше его. Если такого элемента нет, то заменить его на ноль.

**А) Реализация через массив**

**Листинг программы 1:**

print("Ягодяк Павел Сергеевич АИСа-020303-о24")

def next\_greater\_element\_array(arr):

n = len(arr)

result = [0] \* n

stack = []

for i in range(n):

while stack and arr[stack[-1]] < arr[i]:

result[stack.pop()] = arr[i]

stack.append(i)

return result

**Б) Реализация через связный список**

**Листинг программы 2:**

class Node:

def \_\_init\_\_(self, value):

self.value = value

self.next = None

class LinkedList:

def \_\_init\_\_(self):

self.head = None

def append(self, value):

new\_node = Node(value)

if not self.head:

self.head = new\_node

return

last = self.head

while last.next:

last = last.next

last.next = new\_node

def next\_greater\_element\_linked\_list(self):

current = self.head

result = []

stack = []

index = 0

while current:

while stack and stack[-1][0] < current.value:

value, idx = stack.pop()

result[idx] = current.value

stack.append((current.value, index))

result.append(0)

current = current.next

index += 1

return result

**В) Реализация с использованием стандартной библиотеки**

**Листинг программы 3:**

from collections import deque

def next\_greater\_element\_deque(arr):

n = len(arr)

result = [0] \* n

stack = deque()

for i in range(n):

while stack and arr[stack[-1]] < arr[i]:

result[stack.pop()] = arr[i]

stack.append(i)

return result

**Г) Тестирование для сравнения работоспособности и производительности каждой реализации**

**Листинг программы 4:**

import time

from lab3ch1sd import next\_greater\_element\_array

from lab3ch2sd import LinkedList

from lab3ch3sd import next\_greater\_element\_deque

def test\_functions():

    array = [1, 3, 2, 5, 3, 4] \* 10000

    # Тест для массива

    start\_time = time.time()

    result\_array = next\_greater\_element\_array(array)

    end\_time = time.time()

    print("Тест для массива:")

    print(result\_array[:10])  # Печатаем первые 10 элементов для проверки

    print("Время выполнения (массив):", end\_time - start\_time)

    # Тест для связанного списка

    linked\_list = LinkedList()

    for num in array:

        linked\_list.append(num)

    start\_time = time.time()

    result\_linked\_list = linked\_list.next\_greater\_element\_linked\_list()

    end\_time = time.time()

    print("Тест для связанного списка:")

    print(result\_linked\_list[:10])

    print("Время выполнения (связанный список):", end\_time - start\_time)

    # Тест для deque

    start\_time = time.time()

    result\_deque = next\_greater\_element\_deque(array)

    end\_time = time.time()

    print("Тест для deque:")

    print(result\_deque[:10])

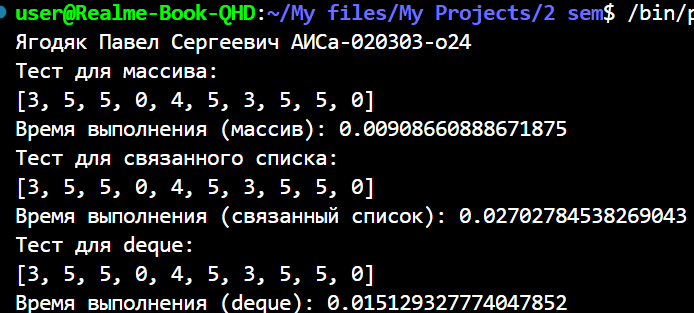
    print("Время выполнения (deque):", end\_time - start\_time)

# Запуск тестов

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

    test\_functions()

**Результат выполнения программы:**



**Вывод:** В результате выполнения лабораторной работы мы изучили три различных подхода к решению одной и той же задачи с использованием различных структур данных. Каждая реализация имеет свои преимущества и недостатки, которые могут влиять на производительность в зависимости от конкретных условий задачи.