Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича»

Факультет иксс

кафедра пиИвт

(СПбГУТ)

ОтчЁт  
ЛАБОРАТОРНАЯ РАБОТА №4

**Разработка программы интроспективной сортировки чисел из файла, используя кольцевую очередь на базе связного списка**

Руководитель,  
старший преподаватель Ерофеев С. А.

подпись, дата

Исполнитель,  
группа ИКПИ-33 Коньков М. Д.

подпись, дата

Постановка задачи

Данная лабораторная работа направлена на реализацию интроспективной сортировки чисел, считанных из файла, с использованием кольцевой очереди на базе связного списка. **Алгоритм**:

1. **Инициализация кольцевой очереди и вспомогательных функций:**

* Программа начинается с определения класса Node, который представляет узел связного списка, и класса CircularQueue, который реализует кольцевую очередь на базе связного списка.
* Затем определяются вспомогательные функции для сортировки: insertion\_sort, partition и introsort\_util.

1. **Чтение чисел из файла и добавление их в очередь:**

* Функция introspective\_sort\_from\_file открывает указанный файл и читает числа из него.
* Каждое прочитанное число добавляется в кольцевую очередь с помощью метода enqueue.

1. **Сортировка чисел с использованием интроспективной сортировки:**

* После того как все числа из файла добавлены в очередь, они извлекаются из нее и сохраняются в виде массива.
* Затем этот массив чисел сортируется с использованием интроспективной сортировки в функции introsort.

1. **Вывод отсортированных чисел на экран:**

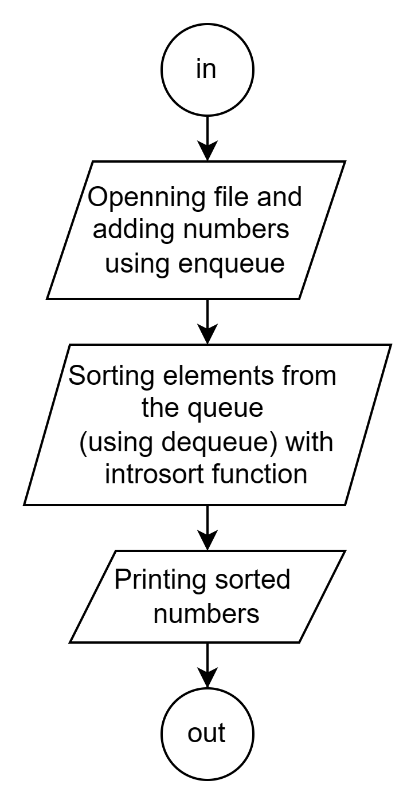
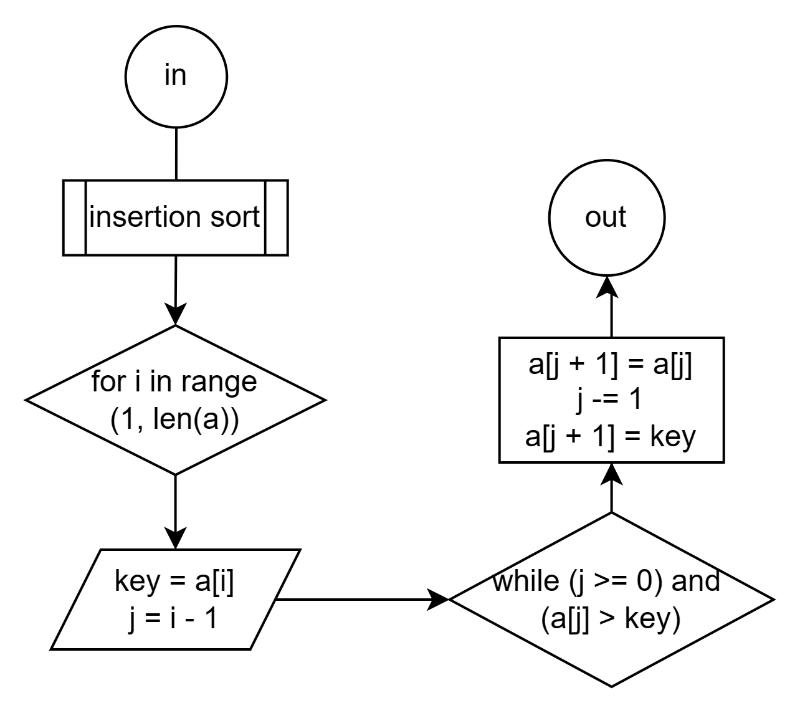
* Отсортированные числа выводятся на экран в порядке возрастания.

1. **Пример использования:**

* В конце программы вызывается функция introspective\_sort\_from\_file с именем файла в качестве аргумента.
* Пользователь может заменить имя файла на свое и выполнить программу для сортировки чисел из другого файла.

**Таким образом, программа осуществляет чтение чисел из файла, их сортировку (при помощи интроспективной сортировки на базе кольцевой очереди), а также вывод итогового отсортированного массива на экран.**

## Общий алгоритм решения: Функция introspective\_sort\_from\_file():

**Алгоритм сортировки вставками: Функция introsort();**

**Быстрая сортировка:**

Алгоритм быстрой сортировки (или quicksort) - это один из наиболее эффективных алгоритмов сортировки. Он основан на стратегии "разделяй и властвуй". Вот как он работает на примере массива:

Предположим, у вас есть массив чисел:

[7, 2, 1, 6, 8, 5, 3, 4]

1. **Выбор опорного элемента**

Выбираем опорный элемент. В данном случае, для простоты, возьмём последний элемент - 4.

2. **Разделение массива**

Проходим по массиву и перемещаем элементы так, чтобы все элементы, меньшие опорного, оказались слева от него, а все элементы, большие или равные опорному, справа от него. В нашем случае, после разделения, массив может выглядеть так:

[2, 1, 3, 4, 8, 5, 7, 6]

3. **Рекурсивная сортировка**

Теперь рекурсивно применяем быструю сортировку к двум подмассивам, левому и правому от опорного элемента. Для каждого из них повторяе шаги 1 и 2.

Для левого подмассива (меньшие элементы):

[2, 1, 3]

Выбираем опорный элемент (последний) - 3. Разделяем массив:

[1, 2, 3]

Для правого подмассива (большие элементы):

[8, 5, 7, 6]

Выбираем опорный элемент (последний) - 6. Разделяем массив:

[5, 6, 7, 8]

4. **Объединение**

После завершения рекурсии объединяем отсортированные подмассивы. Теперь весь массив отсортирован: [1, 2, 3, 4, 5, 6, 7, 8]

**Сортировка вставками:**

Сортировка вставками используется , когда количество элементов небольшое . Это также может быть полезно, когда входной массив почти отсортирован и в большом массиве потеряно лишь несколько элементов.

Исходный массив: [5, 2, 9, 1, 7, 6]

**Шаги алгоритма:**

1. Начинаем со второго элемента (индекс 1) массива. Сравниваем его со всеми предыдущими элементами в упорядоченной части массива и перемещаемся влево, пока не найдем подходящее место для вставки.

[2, 5, 9, 1, 7, 6]

2. Следующий элемент - 9. Сравниваем его с предыдущими элементами в упорядоченной части массива и вставляем его в нужное место.

[2, 5, 9, 1, 7, 6]

3. Элемент 1 меньше всех элементов в упорядоченной части массива. Мы перемещаемся влево и вставляем его на начало упорядоченной части.

[1, 2, 5, 9, 7, 6]

4. Элемент 7 вставляется на свое место в упорядоченной части массива.

[1, 2, 5, 7, 9, 6]

5. Элемент 6 также вставляется на свое место в упорядоченной части массива.

Отсортированный массив: [1, 2, 5, 6, 7, 9]

**Переменные и их назначение:**

* Node - это класс, представляющий узел в связном списке. Каждый узел содержит два основных атрибута: data, который хранит данные, и next, который указывает на следующий узел в списке. Узлы связаны между собой, и каждый узел хранит ссылку на следующий узел в цепочке.
* CircularQueue - это класс, представляющий кольцевую очередь, реализованную с использованием связного списка. Он содержит два основных атрибута: head и tail, которые указывают на начало и конец очереди соответственно.
* num - переменная, используемая для считывания чисел из файла.
* max\_depth - это переменная, которая определяет максимальную глубину рекурсии для интроспективной сортировки.
* threshold - это пороговое значение, используемое для определения, когда следует переключиться с быстрой сортировки на сортировку вставками. Обычно оно вычисляется как квадратный корень из размера массива, умноженный на некоторый множитель (в данном случае 2).
* queue - это объект класса CircularQueue, который используется для временного хранения чисел, считанных из файла, перед их сортировкой. Числа сначала добавляются в эту очередь, а затем извлекаются и сортируются с помощью интроспективной сортировки.
* numbers - это массив, в который временно сохраняются числа после извлечения из очереди queue. После сортировки эти числа будут последовательно выводиться на экран.

Начало формы

**Диапазоны переменных:**

**data, max\_depth, queue, numbers, threshold, num**: Диапазон значений зависит от типа данных, используемого для хранения чисел. Обычно это целые числа, поэтому диапазон будет ограничен значением INT\_MIN до INT\_MAX, где INT\_MIN и INT\_MAX определены в модуле ‘sys’.

## Тестирование:

## Отсортированные числа: 2948 5574 6364 6426 8080 10424 10682 11161 12461 12493 15990 16328 16423 16882 17064 18843 21979 22489 24303 25074 27271 27878 29276 30188 31356 31735 31918 32011 33432 34345 34839 35410 35528 36973 38000 38271 39356 39453 41122 42768 42904 45351 45848 46711 47166 47399 47679 48207 49503 51906 52185 53985 55523 55861 56198 56254 57085 59548 59828 60141 61328 62344 62647 62683 63470 64374 64500 65043 65090 66202 67210 67352 70939 71465 74939 75788 76825 77965 80116 81084 82533 84025 84211 84499 90622 91303 91961 92347 92786 92908 93536 94311 94587 95194 95495 95981 98113 99516 99735 99825 --- [Finished in 76ms]

**Исходный файл numbers.txt:** 27878 35410 2948 56254 36973 55523 45848 62683 65043 34839 92786 29276 75788 47399 99735 91303 10424 51906 93536 30188 39453 99516 84499 32011 76825 39356 80116 17064 77965 81084 95495 95194 5574 62647 94587 63470 53985 67352 70939 18843 35528 16882 31735 11161 59828 16423 55861 95981 74939 27271 6364 21979 38000 59548 15990 52185 84025 56198 91961 57085 10682 98113 66202 82533 65090 41122 46711 92347 38271 94311 71465 64500 92908 8080 60141 99825 45351 12493 34345 22489 31918 47679 31356 16328 84211 64374 62344 33432 67210 49503 61328 6426 47166 48207 42768 12461 90622 24303 25074 42904

## Листинг программы gen.py (Python 3)

import random

# Генерируем список из 1000 случайных чисел

numbers = [random.randint(1, 100000) for \_ in range(100)]

# Записываем числа в файл

with open("numbers.txt", "w") as file:

for number in numbers:

file.write(str(number) + "\n")

## Листинг программы introsort4.py (Python 3)

class Node:

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

self.data = data

self.next = None

class CircularQueue:

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

self.head = None

self.tail = None

def enqueue(self, data):

new\_node = Node(data)

if not self.head:

self.head = new\_node

else:

self.tail.next = new\_node

self.tail = new\_node

self.tail.next = self.head

def dequeue(self):

if not self.head:

return None

data = self.head.data

if self.head == self.tail:

self.head = None

self.tail = None

else:

self.head = self.head.next

self.tail.next = self.head

return data

def is\_empty(self):

return self.head is None

def introsort(arr):

MAX\_DEPTH = 20

threshold = 2 \* int(len(arr) \*\* 0.5)

def insertion\_sort(arr, left, right):

for i in range(left + 1, right + 1):

key = arr[i]

j = i - 1

while j >= left and arr[j] > key:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

def partition(arr, left, right):

pivot = arr[right]

i = left - 1

for j in range(left, right):

if arr[j] <= pivot:

i += 1

arr[i], arr[j] = arr[j], arr[i]

arr[i + 1], arr[right] = arr[right], arr[i + 1]

return i + 1

def introsort\_util(arr, left, right, depth):

if right - left <= threshold:

insertion\_sort(arr, left, right)

return

if depth == 0:

arr[left:right + 1] = sorted(arr[left:right + 1])

return

if left < right:

partition\_index = partition(arr, left, right)

introsort\_util(arr, left, partition\_index - 1, depth - 1)

introsort\_util(arr, partition\_index + 1, right, depth - 1)

introsort\_util(arr, 0, len(arr) - 1, MAX\_DEPTH)

def introspective\_sort\_from\_file(filename):

queue = CircularQueue()

# Read numbers from file and enqueue them

with open(filename, 'r') as file:

for line in file:

for num\_str in line.split():

num = int(num\_str)

queue.enqueue(num)

# Dequeue numbers, sort them using introsort, and enqueue them again

numbers = []

while not queue.is\_empty():

numbers.append(queue.dequeue())

introsort(numbers)

for num in numbers:

print(num)

# Example usage: (file you want to work with)

introspective\_sort\_from\_file("numbers.txt")

**Вывод:**

В ходе выполнения данной лабораторной работы был написан алгоритм интроспективной сортировки на языке Python в среде разработки Sublime Text 3. Было выяснено, что интроспективная сортировка включает в себя выполнение сначала быстрой сортировки (для рекурсивного деления массива на две части) и быстрой сортировки (когда неделимые фрагменты массива должны быть правильно отсортированы) Массив считывался числа из файла numbers.txt. В прорамме использовалась кольцевая очередь на базе связного списка. Функциональные задачи, поставленные перед программой, были успешно выполнены.