САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №2

по курсу «Алгоритмы и структуры данных»

Тема: Сортировка

Выполнил:

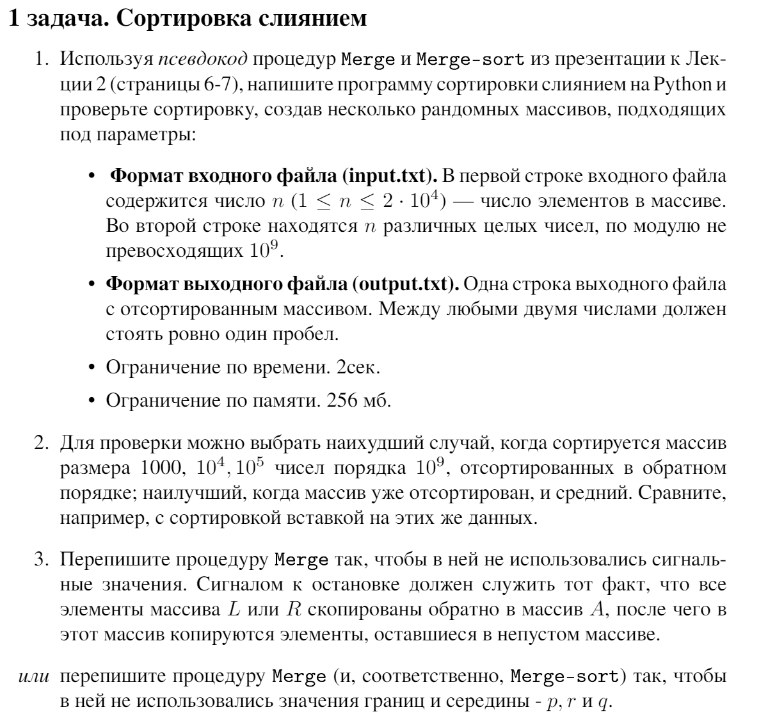
Юшков А.М.

К3139

Проверил:

Санкт-Петербург

2024 г.



Код

def merge\_sort(a):  
 if len(a) > 1:  
 mid = len(a) // 2  
 left\_half = a[:mid]  
 right\_half = a[mid:]  
 merge\_sort(left\_half)  
 merge\_sort(right\_half)  
 i = j = k = 0  
 while i < len(left\_half) and j < len(right\_half):  
 if left\_half[i] < right\_half[j]:  
 a[k] = left\_half[i]  
 i += 1  
 else:  
 a[k] = right\_half[j]  
 j += 1  
 k += 1  
 while i < len(left\_half):  
 a[k] = left\_half[i]  
 i += 1  
 k += 1  
 while j < len(right\_half):  
 a[k] = right\_half[j]  
 j += 1  
 k += 1  
with open("input.txt", "r") as file:  
 n = int(file.readline())  
 arr = [int(i) for i in file.readline().split()]  
merge\_sort(arr)  
print(arr)

Пояснение к коду:

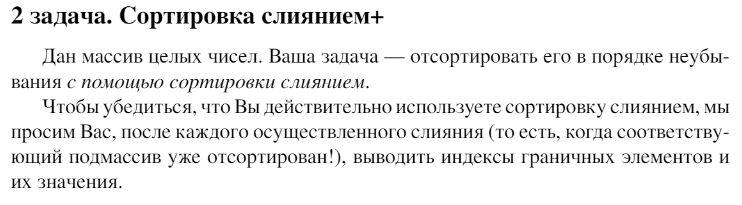
1)Считываем из файла данные

2)Разделяем пассив на левую и правую части

3)рекурсивно сортируем левые и правые части

4)Заполняем массив

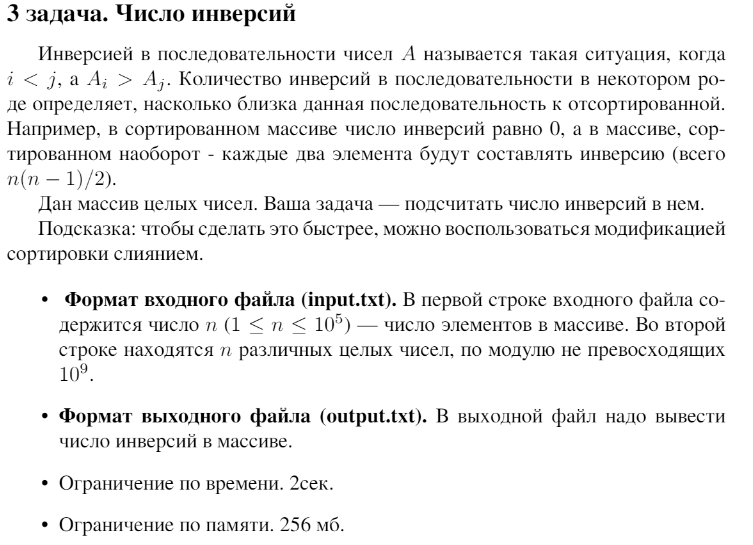
Вывод по задаче: В ходе выполнения задачи я узнал как происходит сортировка слиянием, вспомнил циклы, функции, условные операторы и методы ввода-вывода



Код

def merge(left, right):  
 merged = []  
 i = j = 0  
 while i < len(left) and j < len(right):  
 if left[i] <= right[j]:  
 merged.append(left[i])  
 i += 1  
 else:  
 merged.append(right[j])  
 j += 1  
 merged.extend(left[i:])  
 merged.extend(right[j:])  
 return merged  
  
def merge\_sort(arr):  
 if len(arr) <= 1:  
 return arr  
 mid = len(arr) // 2  
 left = merge\_sort(arr[:mid])  
 right = merge\_sort(arr[mid:])  
 merged = merge(left, right)  
 print("Слияние:", left, right)  
 print("Значение:", merged)  
 return merged  
  
with open("input.txt", "r") as file:  
 n = int(file.readline())  
 a = [int(i) for i in file.readline().split()]  
a = merge\_sort(a)  
print(a)

Пояснение к коду: В отличие от предыдущей задачи функция сортировки левой и правой части вынесена отдельно, выводится промежуточные шаги сортировки



Код:

def merge\_and\_count(arr, temp\_arr, left, mid, right):  
 i = left  
 j = mid + 1  
 k = left  
 inv\_count = 0  
 while i <= mid and j <= right:  
 if arr[i] <= arr[j]:  
 temp\_arr[k] = arr[i]  
 i += 1  
 else:  
 temp\_arr[k] = arr[j]  
 inv\_count += (mid - i + 1)  
 j += 1  
 k += 1  
 while i <= mid:  
 temp\_arr[k] = arr[i]  
 i += 1  
 k += 1  
 while j <= right:  
 temp\_arr[k] = arr[j]  
 j += 1  
 k += 1  
 for i in range(left, right + 1):  
 arr[i] = temp\_arr[i]  
 return inv\_count  
def merge\_sort\_and\_count(arr, temp\_arr, left, right):  
 inv\_count = 0  
 if left < right:  
 mid = (left + right) // 2  
 inv\_count += merge\_sort\_and\_count(arr, temp\_arr, left, mid)  
 inv\_count += merge\_sort\_and\_count(arr, temp\_arr, mid + 1, right)  
 inv\_count += merge\_and\_count(arr, temp\_arr, left, mid, right)  
 return inv\_count  
with open("input.txt", "r") as file:  
 n = int(file.readline())  
 a = [int(i) for i in file.readline().split()]  
n = len(a)  
temp\_array = [0] \* n  
result = merge\_sort\_and\_count(a, temp\_array, 0, n - 1)  
print("Количество инверсий:", result)

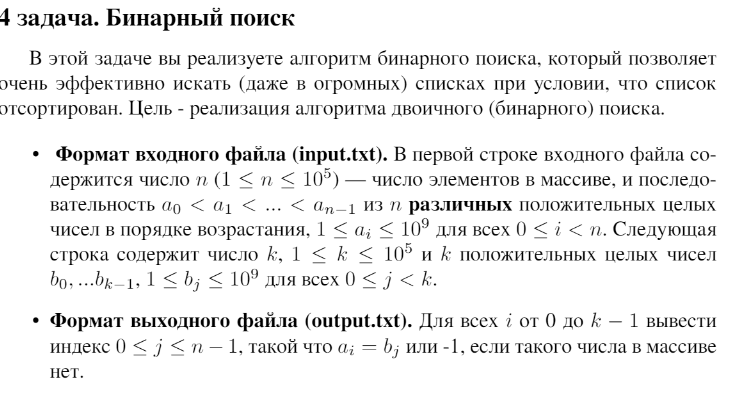
Пояснение к коду:

1)Считывем данные из файла

2)Вызываем функцию, ей на вход подается левая и правая границы и два массива

3)Сортируем массив и считаем количество инверсий

Вывод по задаче: В ходе выполнения задачи я узнал как происходит сортировка слиянием, как посчитать кол-во инверсий, вспомнил циклы, функции, условные операторы и методы ввода-вывода



Код

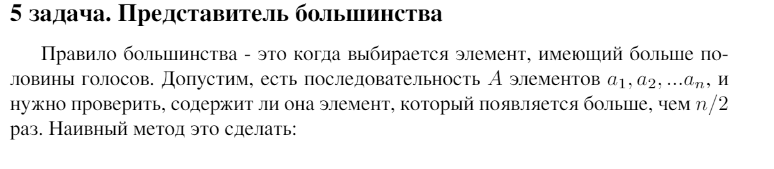
def binary\_search(arr, target):  
 left, right = 0, len(arr) - 1  
 while left <= right:  
 mid = (left + right) // 2  
 if arr[mid] == target:  
 return mid  
 elif arr[mid] < target:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return -1  
  
with open("input.txt", "r") as file:  
 n1 = int(file.readline())  
 a = [int(i) for i in file.readline().split()]  
 n2 = int(file.readline())  
 b = [int(i) for i in file.readline().split()]  
c = []  
for i in range(len(b)):  
 c.append(binary\_search(a, b[i]))  
print(c)

Пояснение к коду

1)Считываем данный из файла

2)запускаем цикл поиска для каждого элемента массива

3)С помощью алгоритма бинарного поиска ищем число в массиве



import time, psutil, random  
t\_start = time.perf\_counter()  
def majority\_element(nums):  
 candidate = None  
 count = 0  
 for num in nums:  
 if count == 0:  
 candidate = num  
 count = 1  
 elif num == candidate:  
 count += 1  
 else:  
 count -= 1  
 if candidate is not None and nums.count(candidate) > len(nums) // 2:  
 return True  
 else:  
 return False  
  
with open("input.txt", "r") as file:  
 n = int(file.readline())  
 a = [int(i) for i in file.readline().split()]  
if majority\_element(a):  
 print('1')  
else:  
 print('0')

Пояснение

1)Считываем данный из файла

2)Ищем кандидата

3)Проверяем является ли предположительный кандидат кандидатом

**Вывод**

 В ходе выполнения лабораторной работы были изучены методы сортировки слиянием, бинарный поиск и поиск представителя большинства