**Министерство науки и высшего образования Российской Федерации**
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**
**(Университет ИТМО)**

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Интеллектуальные системы в гуманитарной сфере**

Направление подготовки **45.03.04 Интеллектуальные системы в гуманитарной сфере**

О Т Ч Е Т

**лабораторной работе 2**

на тему: “Сортировка слиянием. Метод декомпозиции”

Обучающийся (или несколько) ФИО, № группы
Королева Екатерина
К3143

Работа выполнена с оценкой ____

Преподаватель:

_____
(подпись)

Дата 08.10.2021

Санкт-Петербург, 2021

1 задача. Сортировка слиянием.

```
import time #импортировать модуль time
import random #импортировать модуль random
t_start = time.perf_counter() #вызов функции для измерения
процессорного времени
from collections import deque #импортировать модуль
deque(очередь)


def merge_sort(arr): #логика функции
  length = len(arr) #переменной lenth присвоить значение длины
переменнной arr
  if length == 1: #если lenth равно 1, то...
     return arr #верни arr: сортировать единичный массив нет
смысла:)
  else: #а вот если массив не единичный, то...
     mid = length // 2 #переменной mid присвоить значение
половины переменной length
     left = merge_sort(arr[:mid]) #left присвоить левую часть
массива merge_sort (если длина массива нечетная, то "серединка"
попадет сюды)
     right = merge_sort(arr[mid:]) #right присвоить правую часть
массива merge_sort
     left = deque(left) #в left вызвать начальный элемент массива
     right = deque(right) #в right вызвать конечный элемент
массива
     i = len(left) #переменной i присвоить длину left
     j = len(right) #переменной j присвоить длину right
     left.append(float('inf')) #добавить бесконечность слева (по
условию)
     right.append(float('inf')) #добавить бесконечность справа
(по условию)
     arr = [] #обнулить массив (мы все сделали, проверили, а
теперь обманули систему, чтобы в уже использованный массив
положить новый результат. magic)
     while i > 0 or j > 0: #в цикле while делаем следующее: пока
переменные i и j больше нуля:
        if left[0] < right[0]: #если левый массив меньше правого,
то...
           arr.append(left.popleft()) #в массив arr вложить
массив left
           i -= 1 #i уменьшить на 1
        else: #иначе...
           arr.append(right.popleft()) #в массив arr вложить
массив right
           j -= 1 #j уменьшить на 1
     return arr #вернуть arr. В чем суть: мы смотрим, какой из
отсортированных массивов меньше, и добавляем сначала меньший в
arr, а потом больший
```

```
with open('input.txt', 'w') as f: #тут дальше все понятно, и мне
лень писать)
  for i in range(1): #
     a = f.write(str(random.randint(1, 1000))) #
     f.write(' ') #

if __name__ == '__main__': #
   f = open('input.txt') #
   a = list(map(int, f.readline().split())) #
   a = merge_sort(a) #

print(a) #
print("Время работы: %s секунд" % (time.perf_counter() -
t_start)) #
```

2 задача. Сортировка слиянием+

```
import time
t_start = time.perf_counter()
from math import inf


def merge(arr: list, p: int, q: int, r: int):
   arr.insert(0, -inf)
   n1 = q - p + 1
   n2 = r - q
   left = [-inf]
   right = [-inf]
   for i in range(1, n1+1):
       left.append(arr[p+i-1])
   for j in range(1, n2+1):
       right.append(arr[q+j])
   left.append(inf)
   right.append(inf)
   i, j = 1, 1
   for k in range(p, r+1):
       if left[i] <= right[j]:
           arr[k] = left[i]
           i = i + 1
       else:
           arr[k] = right[j]
           j = j + 1
   del arr[0]
   print(p, r, arr[p-1], arr[r-1])


def merge_sort(a, p=None, r=None):
   if p is None:
       p = 1
```

```
    if r is None:
        r = len(a)
    if p >= r:
        return a
    q = (p + r)//2
    merge_sort(a, p, q)
    merge_sort(a, q+1, r)
    merge(a, p, q, r)
    return a


print(merge_sort([1, 8, 2, 1, 4, 7, 3, 2, 3, 6]))
print("Время работы: %s секунд" % (time.perf_counter() -
t_start))
```

3 задача. Число инверсий

```
import time
t_start = time.perf_counter()
from collections import deque
from math import inf


def merge_sort(arr):
    global count
    length = len(arr)
    if length == 1:
        return arr
    mid = length // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])
    left = deque(left)
    right = deque(right)
    i = len(left)
    j = len(right)
    left.append(inf)
    right.append(inf)
    arr = []
    while i > 0 or j > 0:
        if left[0] <= right[0]:
            arr.append(left.popleft())
            i -= 1
        else:
            arr.append(right.popleft())
            j -= 1
            if len(left) > 1:
                count += len(left) - 1
    return arr
```

```
count = 0
a = [1, 8, 2, 1, 4, 7, 3, 2, 3, 6]
a = merge_sort(a)
print(count)
print("Время работы: %s секунд" % (time.perf_counter() -
t_start))
```

4 задача. Бинарный поиск

```
import time
t_start = time.perf_counter()
def binary_search(arr, x, left=None, right=None):
    if left is None:
        left = 0
    if right is None:
        right = len(arr) - 1
    if right < left:
        return -1
    mid = (right - left)//2 + left
    if mid >= len(arr):
        return -1
    if x == arr[mid]:
        return mid
    elif x < arr[mid]:
        return binary_search(arr, x, left=left, right=mid-1)
    else:
        return binary_search(arr, x, left=mid+1, right=right)


n = int(input())
a = sorted(map(int, input().split()))
k = int(input())
for i in map(int, input().split()):
    b = binary_search(a, i)
    print(b, end=' ')
print("Время работы: %s секунд" % (time.perf_counter() -
t_start))
```

5 задача. Представитель большинства

```
import time
t_start = time.perf_counter()
from math import inf
```

```
from collections import deque

def merge_sort(arr):
  length = len(arr)
  if length == 1:
     return arr
  else:
     mid = length // 2
     left = merge_sort(arr[:mid])
     right = merge_sort(arr[mid:])
     left = deque(left)
     right = deque(right)
     i = len(left)
     j = len(right)
     left.append(inf)
     right.append(inf)
     arr = []
     while i > 0 or j > 0:
        if left[0] < right[0]:
           arr.append(left.popleft())
           i -= 1
        else:
           arr.append(right.popleft())
           j -= 1
     return arr

def major_exist(arr: list) -> bool:
   arr = merge_sort(arr)
   length = len(arr)
   mid = length // 2
   left = arr.index(arr[mid])
   right = length - arr[::-1].index(arr[mid]) - 1
   if right - left + 1 > length/2:
       return True
   return False


n = int(input())
a = list(map(int, input().split()))
print(int(major_exist(a)))
print("Время работы: %s секунд" % (time.perf_counter() -
t_start))
```