

Лабораторная работа 1 - Сортировки и линейный поиск

Цель

Реализовать базовые алгоритмы сортировки и поиска, а также проверить соответствие ограничениям по времени (1 сек) и памяти (256 МБ).

Задания и решения

Задание 1 - Сортировка вставкой (Insertion sort)

Реализована классическая сортировка вставками: для каждого элемента `a[i]` выполняется сдвиг больших элементов влево и вставка `key` на нужную позицию.

- Время: $O(n^2)$
- Память: $O(1)$
- Ввод: `n`, затем `n` чисел
- Вывод: отсортированный массив через пробел



```
ex1.py  X
n1 > ex1.py > insertion_sort
1 def insertion_sort(a, n):
2     for i in range(1, n):
3         key = a[i]
4         j = i - 1
5         while j >= 0 and a[j] > key:
6             a[j + 1] = a[j]
7             j -= 1
8             a[j + 1] = key
9
10
11 with open("n1/input.txt", "r", encoding="utf-8") as f:
12     n = int(f.readline())
13     a = list(map(int, f.readline().split()))
14
15 insertion_sort(a, n)
16
17 with open("n1/output.txt", "w", encoding="utf-8") as f:
18     f.write(" ".join(map(str, a)))
19

input.txt  X
n1 > input.txt
1 6
2 123 2 34 96 55 9

output.txt  X
n1 > output.txt
1 2 9 34 55 96 123
```

Задание 4 - Линейный поиск (Linear search)

Выполняется сканирование массива слева направо и сбор индексов всех совпадений `V`.

Формат вывода объединённый:

- если совпадений нет - `-1`
- если ровно одно совпадение - печатается только индекс
- если совпадений 2 и более - печатается `k`, затем на новой строке индексы через запятую

Индексация: с 1.

Пример (2 совпадения):

```
ex2.py  output.txt
n2 > ex2.py > ...
1 with open("n2/input.txt", "r", encoding="utf-8") as f:
2     lines = [line.rstrip("\n") for line in f.readlines()]
3
4 a = list(map(int, lines[0].split())) if len(lines) > 0 and lines[0].strip() else []
5 v = int(lines[1].strip()) if len(lines) > 1 else 0
6
7 idx = []
8 for i, x in enumerate(a, start=1):
9     if x == v:
10         idx.append(i)
11
12 with open("n2/output.txt", "w", encoding="utf-8") as f:
13     if len(idx) == 0:
14         f.write("-1")
15     elif len(idx) == 1:
16         f.write(str(idx[0]))
17     else:
18         f.write(str(len(idx)) + "\n")
19         f.write(".".join(map(str, idx)))
20
```

```
input.txt
n2 > input.txt
1 1 2 4 4 5 7 29 52
2 4

output.txt
n2 > output.txt
1 2
2 3,4
```

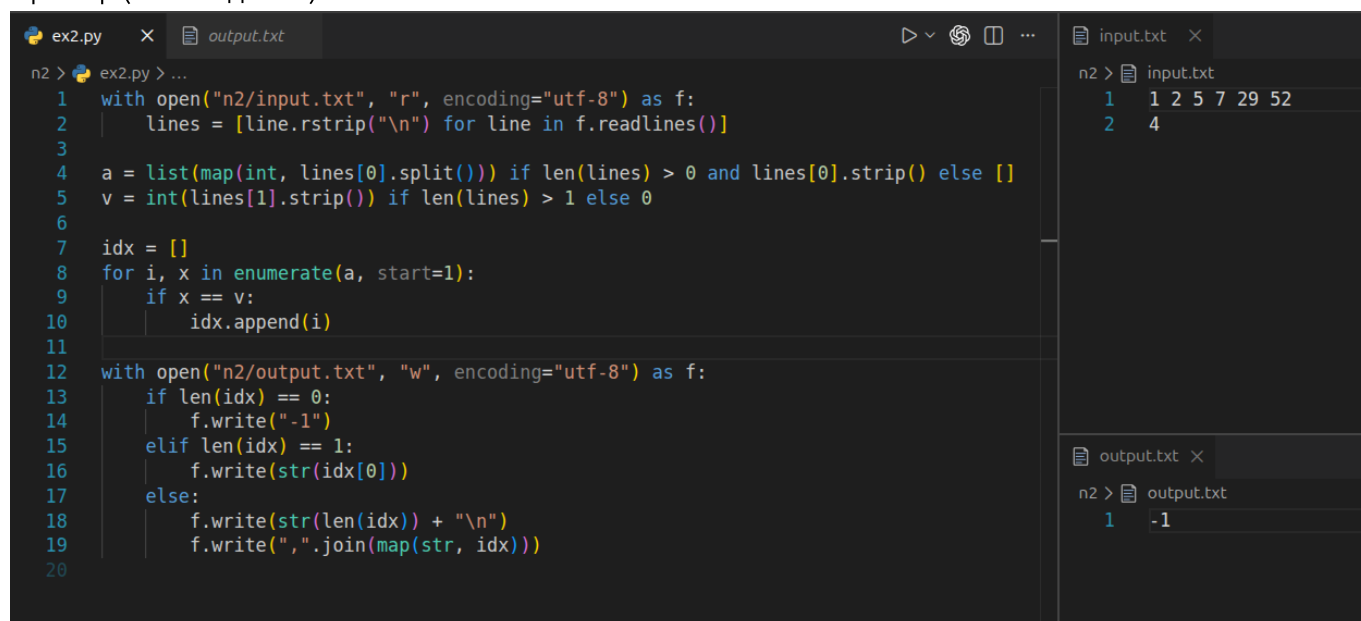
Пример (1 совпадение):

```
ex2.py  output.txt
n2 > ex2.py > ...
1 with open("n2/input.txt", "r", encoding="utf-8") as f:
2     lines = [line.rstrip("\n") for line in f.readlines()]
3
4 a = list(map(int, lines[0].split())) if len(lines) > 0 and lines[0].strip() else []
5 v = int(lines[1].strip()) if len(lines) > 1 else 0
6
7 idx = []
8 for i, x in enumerate(a, start=1):
9     if x == v:
10         idx.append(i)
11
12 with open("n2/output.txt", "w", encoding="utf-8") as f:
13     if len(idx) == 0:
14         f.write("-1")
15     elif len(idx) == 1:
16         f.write(str(idx[0]))
17     else:
18         f.write(str(len(idx)) + "\n")
19         f.write(".".join(map(str, idx)))
20
```

```
input.txt
n2 > input.txt
1 1 2 4 5 7 29 52
2 4

output.txt
n2 > output.txt
1 3
```

Пример (0 совпадений):



The screenshot shows a code editor with two tabs: 'ex2.py' and 'output.txt'. The 'ex2.py' tab contains the following Python code:

```
n2 > ex2.py > ...
1 with open("n2/input.txt", "r", encoding="utf-8") as f:
2     lines = [line.rstrip("\n") for line in f.readlines()]
3
4 a = list(map(int, lines[0].split())) if len(lines) > 0 and lines[0].strip() else []
5 v = int(lines[1].strip()) if len(lines) > 1 else 0
6
7 idx = []
8 for i, x in enumerate(a, start=1):
9     if x == v:
10         idx.append(i)
11
12 with open("n2/output.txt", "w", encoding="utf-8") as f:
13     if len(idx) == 0:
14         f.write("-1")
15     elif len(idx) == 1:
16         f.write(str(idx[0]))
17     else:
18         f.write(str(len(idx)) + "\n")
19         f.write(",".join(map(str, idx)))
20
```

The 'output.txt' tab shows the input data:

```
n2 > input.txt
1 1 2 5 7 29 52
2 4
```

Below the input data, the output of the script is shown:

```
n2 > output.txt
1 -1
```

Задание 8 - Секретарь Своп

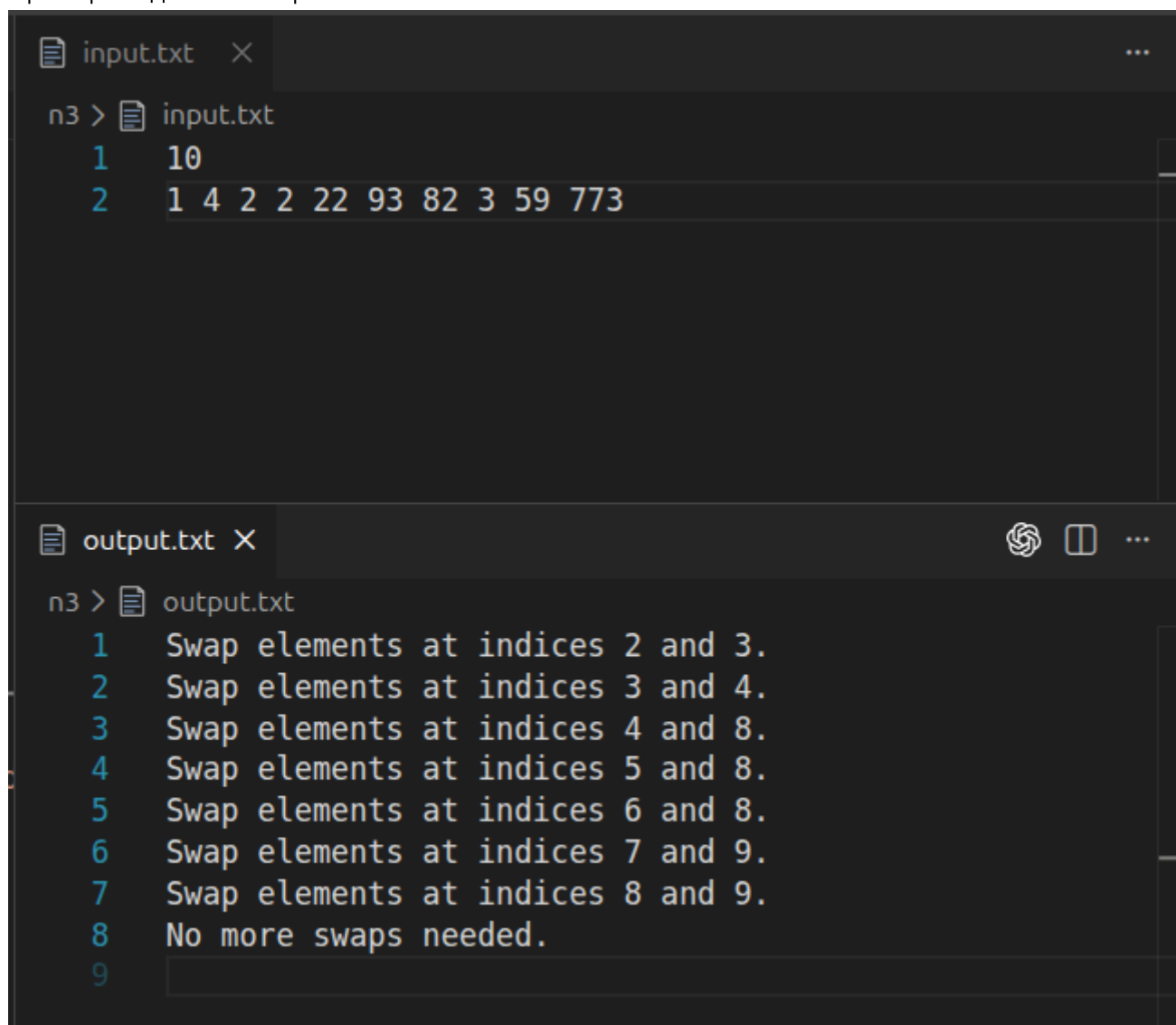
Нужно отсортировать массив по неубыванию и вывести все произведённые перестановки в формате:
Swap elements at indices X and Y.

После всех перестановок: **No more swaps needed.**

Чтобы не генерировать слишком длинный лог, используется подход с выбором минимального элемента на каждом шаге (по сути selection-sort):

- максимум перестановок: **n - 1**
- лог пишется сразу в файл (без накопления строк в памяти)

Пример входа и лога перестановок:



The screenshot shows a terminal window with two tabs: 'input.txt' and 'output.txt'. The 'input.txt' tab is active, showing the following content:

```
n3 > input.txt
1 10
2 1 4 2 2 22 93 82 3 59 773
```

The 'output.txt' tab is also visible, showing the following content:

```
n3 > output.txt
1 Swap elements at indices 2 and 3.
2 Swap elements at indices 3 and 4.
3 Swap elements at indices 4 and 8.
4 Swap elements at indices 5 and 8.
5 Swap elements at indices 6 and 8.
6 Swap elements at indices 7 and 9.
7 Swap elements at indices 8 and 9.
8 No more swaps needed.
9
```

Код

```
import time
import tracemalloc

def secretary_swap_sort_with_log(a, n, out):
    for i in range(n):
        min_idx = i
        for j in range(i + 1, n):
            if a[j] < a[min_idx]:
                min_idx = j

        if min_idx != i:
            x, y = i + 1, min_idx + 1
            if x > y:
                x, y = y, x
            out.write(f"Swap elements at indices {x} and {y}.\n")
            a[i], a[min_idx] = a[min_idx], a[i]
```

```
out.write("No more swaps needed.\n")

def bytes_to_mb(b):
    return b / (1024 * 1024)

if __name__ == "__main__":
    with open("n3/input.txt", "r", encoding="utf-8") as f:
        n = int(f.readline())
        a = list(map(int, f.readline().split()))

    tracemalloc.start()
    t0 = time.perf_counter()

    with open("n3/output.txt", "w", encoding="utf-8") as out:
        secretary_swap_sort_with_log(a, n, out)

    t1 = time.perf_counter()
    current, peak = tracemalloc.get_traced_memory()
    tracemalloc.stop()

    elapsed = t1 - t0

    print(f"Time: {elapsed:.6f} sec\n")
    print(f"Peak memory: {peak} bytes ({bytes_to_mb(peak):.3f} MB)\n")

    print("Limits: 1 sec, 256 MB\n")
```

Время и память для кода выше

```
● chikirao@chikirao-MCLG-XX:~/main/itmo/algo/lab1$
go/lab1/n3/ex3.py
Time: 0.000790 sec

Peak memory: 5941 bytes (0.006 MB)

Limits: 1 sec, 256 MB
```