Лабораторная работа №3

Тема: Алгоритмический мини-пакет (структуры данных и сортировки) Ограничения

- Запрещено: list.sort(), sorted() в реализации сортировок.
- Разрешено: стандартная библиотека (в Medium допускается functools.cmp_to_key).
- Bucket sort: по умолчанию сортируем float в [0,1) (или пишем нормализацию).
- Структуры данных должны выбрасывать исключения (ValueError или IndexError) при некорректных операциях.

Варианты заданий

Факториал и Фибоначчи. Необходимо реализовать функции, которые вычисляют функции факториала и фибоначчи.

```
def factorial(n: int) -> int: ...
def factorial_recursive(n: int) -> int: ...
def fibo(n: int) -> int: ... def fibo_recursive(n: int) -> int: ...
```

Практика:

- LeetCode 509 Fibonacci Number – https://leetcode.com/problems/fibonaccinumber/description/
- LeetCode 70 Climbing Stair – https://leetcode.com/problems/climbing-stairs/description/

Сортировки пузырьковая, быстрая, count sort, radix sort, bucket sort

```
1. def bubble_sort(a: list[int]) -> list[int]:      ...
2. def quick_sort(a: list[int]) -> list[int]:      ...
3. def counting_sort(a: list[int]) -> list[int] :      ...
4. def radix_sort(a: list[int], base: int = 10) -> list[int]:
       ...
5. def bucket_sort(a: list[float], buckets: int | None = None) -> list[float]:
       ...
6. def heap_sort(a: list[int]) -> list[int]:
       ...
```

Практика:

- LeetCode 75 Sort Colors – https://leetcode.com/problems/sort-colors/description/
- Hackerrank CountingSort 1 – https://www.hackerrank.com/challenges/countingsort1/problem
- Hackerrank CountingSort 2– https://www.hackerrank.com/challenges/countingsort2/problem
- Hackerrank CountingSort 3– https://www.hackerrank.com/challenges/countingsort3/problem
- Hackerrank QuickSort 1 – https://www.hackerrank.com/challenges/quicksort1/problem
- Top K Frequent Elements – https://leetcode.com/problems/top-k-frequentelements/description/
- K-th Largest Element in Array – https://leetcode.com/problems/kth-largest-element-in-anarray/description/

Структуры данных (На выбор):

1. Стек:
   a. Связный список (Node(value, next))

b. Стек на list
c. Стек на очередях

```
class Stack:                    def push(self, x: int) -> None: ...
                def pop(self) -> int: ...      # исключение при пустом                    def peek(self) -> int: ...
# исключение при пустом                 def is_empty(self) -> bool: ...                def __len__(self) -> int: ...
                                        def min(self) -> int: ... # за константу для Medium
```

2. Очередь
   a. Связный список
   b. Очередь на list
   c. Очередь на стеках

```
class Queue:
    def enqueue(self, x: int) -> None: ...
    def dequeue(self) -> int: ...    # исключение при пустой    def front(self) -> int: ...    #
исключение при пустой    def is_empty(self) -> bool: ...    def __len__(self) -> int: ...
```

Практика:

- LeetCode 20 Valid Parentheses – https://leetcode.com/problems/validparentheses/description/
- LeetCode 225 Implement Stack using Queues – https://leetcode.com/problems/implementstack-using-queues/description/
- LeetCode 232 Implement Queue using Stacks – https://leetcode.com/problems/implementqueue-using-stacks/description/

Обвесы (На выбор):

1. Генерация тест-кейсов

```
def rand_int_array(n: int, lo: int, hi: int, *, distinct=False, seed=None) -> list[int]: ... def nearly_sorted(n: int, swaps: int, *, seed=None) ->
list[int]: ... def many_duplicates(n: int, k_unique=5, *, seed=None) -> list[int]: ...
def reverse_sorted(n: int) -> list[int]: ...
def rand_float_array(n: int, lo=0.0, hi=1.0, *, seed=None) -> list[float]: ...
```

2. Бенчмарк `def timeit_once(func, *args, **kwargs) -> float: ...`
   `def benchmark_sorts(arrays: dict[str, list], algos: dict[str, callable]) -> dict[str, dict[str, float]]: ...`
3. Поддержка ключей и компаратора в сортировках `...(a: list[T], key: Callable[[T], Any] | None = None,`
   `cmp: Callable[[T, T], int] | None = None) -> list[T]`

Критерии оценивания

- Корректность– 40
- Сортировки (Easy) – 20
- Структуры данных (Easy) – 30
- Генераторы и тайминг (Medium) – 5
- Ключи/компаратор (Medium) – 5

Дополнительно: CLI – +3%, отчёт с бенчмарками – +2%, решенная практика: +5%.