Звіт
Лабораторна робота 2
ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ
Большаков Андрій МІТ-31

Лабораторна робота №2. Технології програмування

9*	**3**	Реалізувати структуру даних відсортований масив без **дублікатів**. Масив завжди підтримується у відсортованому стані. При додаванні елемента він вставляється у вже відсортований масив, масив після вставки не сортується! Доступні операції: додати та видалити елемент/елементи, очистити, злити з іншим масивом.

https://github.com/Utilka/univ_programming_technologies_proj

проанализировав обьект я делаю вывод о целесообразности таких тестов:

тест основных функций (создание массива, добавление и сортировка элементов)

```
def test_sorting(self):
    self.cases = ([], [0], [1], list(range(0, 10)), list(range(10, 0, -1)),
                  [], [-0], [-1], list(range(-10, 0)), list(range(0, -10, -1)))

    self.results = ([], [0], [1], list(range(0, 10)), list(range(1, 11)),
                    [], [0], [-1], list(range(-10, 0)), list(range(-9, 1)))

    self.common_result_eq_test(self.cases, self.results, 'test_sorting')
```

тест сортировки в обратном порядке

```
def test_reverse_sorting(self):
    self.cases = ([], [0], [1], list(range(0, 10)), list(range(10, 0, -1)),
                  [], [-0], [-1], list(range(-10, 0)), list(range(0, -10, -1)))

    self.results = ([], [0], [1], list(range(9, -1, -1)), list(range(10, 0, -1)),
                    [], [0], [-1], list(range(-1, -11, -1)), list(range(-1, -11, -1)))

    self.common_result_eq_test(self.cases, self.results, 'test_reverse_sorting',
ascending_order=False)
```

тест удаления дубликатов

```
def test_duplicates_removal(self):
    self.cases = ([1, 1], [1, 1, 2, 2, 3, 3, 4, 4],
                  [-1, -1], [-1, -1, -2, -2, -3, -3, -4, -4],
                  list(range(0, 10)) + list(range(0, 10)))

    self.results = ([1], [1, 2, 3, 4],
                    [-1], [-4, -3, -2, -1],
                    list(range(0, 10)))

    self.common_result_eq_test(self.cases, self.results, 'test_duplicates_removal')
```

тест универсальности (сортировка не только целых чисел)

```
def test_universality(self):
    self.cases = ([4, 2, 8], list('abczzefgaa'), [True, False],
                  [float(i) / 10 for i in range(10, 0, -1)],
                  [[6, 7], [3, 4, 5], [3, 4], [2], [1, 2]])

    self.results = ([2, 4, 8], list('abcefgz'), [False, True],
                    [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
                    [[1, 2], [2], [3, 4], [3, 4, 5], [6, 7]])

    self.common_result_eq_test(self.cases, self.results, 'test_universality')
```

тест слития с другим отсортированым массивом

```
def test_merge(self):
    self.cases = ((list(range(0, 5)), list(range(5, 10))),
                  (list(range(0, 5)), list(range(3, 8))),
                  (list(range(0, 5)), list(range(0, 5))),
                  (list(range(0, 5)), list(range(-5, 0))),
                  (list(range(0, 5)), list(range(-3, 2))))

    self.results = (list(range(0, 10)),
                    list(range(0, 8)),
                    list(range(0, 5)),
                    list(range(-5, 5)),
                    list(range(-3, 5)))

    for a, r in zip(self.cases, self.results):
        with self.subTest(case=a):
            b = SortedListNoDubl(list(a[0]))
            c = SortedListNoDubl(list(a[1]))
            b.merge_with_sorted_list(c)
            self.assertTrue(all(x == y for x, y in zip(b.get_list(), r)),
                            msg="test_merge : elements changed. a = " + str(a) + ", b = " + str(b))
```

тест поднятия ошибки в случае добавления несравнимых объектов

```
def test_exceptions(self):
    self.cases = (([1, 2], "a"),
                  ([1, 2], [1, 2]))

    for a in self.cases:
        with self.subTest(case=a):
            b = SortedListNoDubl(list(a[0]))
            self.assertRaises(TypeError, b.add, a[1])
```

в многих из этих тестов используется функция `common_result_eq_test`

```
def common_result_eq_test(self, cases, results, test="",ascending_order=True):
   for a, r in zip(cases, results):
      with self.subTest(case=a):
         b = SortedListNoDubl(list(a),ascending_order=ascending_order)
         self.assertTrue(all(x == y for x, y in zip(b.get_list(), r)),
                         msg=test + ": elements changed. a = " + str(a) + ", b = " + str(b) + ", r = "
+ str(r))
```

принцип ее работы прост, она берет данные случая, прогоняет из через создание объекта и затем сравниевает полученый результат с заранее заданым

результаты запуска тестов:

```
utilka@utilka-Inspiron-3580:~/Personal/productivity_stuff/univer/sem_5/
Programming technologies/univ_programming_technologies_proj$ python3 unitTest.py
test_sorting (__main__.TestArr) ... ok
test_reverse_sorting (__main__.TestArr) ... test_duplicates_removal
(__main__.TestArr) ... ok
test_universality (__main__.TestArr) ... ok
test_merge (__main__.TestArr) ... test_exceptions (__main__.TestArr) ... ok

======================================================================
ERROR: test_merge (__main__.TestArr) (case=([0, 1, 2, 3, 4], [5, 6, 7, 8, 9]))
----------------------------------------------------------------------
Traceback (most recent call last):
  File "unitTest.py", line 72, in test_merge
    b.merge_with_sorted_list(c)
  File "/home/utilka/Personal/productivity_stuff/univer/sem_5/Programming
technologies/univ_programming_technologies_proj/sortedListNoDubl.py", line 35,
in merge_with_sorted_list
    if self.__array[ind] == ad_item:
IndexError: list index out of range

======================================================================
ERROR: test_merge (__main__.TestArr) (case=([0, 1, 2, 3, 4], [3, 4, 5, 6, 7]))
----------------------------------------------------------------------
Traceback (most recent call last):
  File "unitTest.py", line 72, in test_merge
    b.merge_with_sorted_list(c)
  File "/home/utilka/Personal/productivity_stuff/univer/sem_5/Programming
technologies/univ_programming_technologies_proj/sortedListNoDubl.py", line 35,
in merge_with_sorted_list
    if self.__array[ind] == ad_item:
IndexError: list index out of range

======================================================================
FAIL: test_reverse_sorting (__main__.TestArr) (case=[0, -1, -2, -3, -4, -5, -6,
-7, -8, -9])
----------------------------------------------------------------------
Traceback (most recent call last):
  File "unitTest.py", line 12, in common_result_eq_test
    self.assertTrue(all(x == y for x, y in zip(b.get_list(), r)),
AssertionError: False is not true : test_reverse_sorting: elements changed. a =
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9], b = [0, -1, -2, -3, -4, -5, -6, -7, -8,
-9], r = [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]

----------------------------------------------------------------------
```

```
Ran 6 tests in 0.001s

FAILED (failures=1, errors=2)
```

из результатов видим что мы имеем проблему с слитием массивов, а также с одним из тестов обратной сортировки

в провалившемся тесте обратной сортировки был некоректно задан результат сортировки, ошибка не в программе а в тесте

в тесте слития массивов видим проверку с индексацией при проверке присутствия елемента в массиве, я решил использовать строку которая бы сэкономила бы нам проверку присутсвия елмента в целом списке, решил заменить ее на более простую для читания и не подверженую таким ошибкам “<element> in <list>”

после внесенных изменнений снова запускаем тесты

```
utilka@utilka-Inspiron-3580:~/Personal/productivity_stuff/univer/sem_5/
Programming technologies/univ_programming_technologies_proj$ python3 unitTest.py

test_sorting (__main__.TestArr) ... ok
test_reverse_sorting (__main__.TestArr) ... ok
test_duplicates_removal (__main__.TestArr) ... ok
test_universality (__main__.TestArr) ... ok
test_merge (__main__.TestArr) ... ok
test_exceptions (__main__.TestArr) ... ok

----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
```

успех

Выводы:
Тестировние позволияет отловить ошибки в коде и случаи некорректной работы, как и было продемонстрировано в ходе выполнения работы