Львівський національний університет імені Івана Франка
Факультет електроніки та комп’ютерних технологій

Звіт про виконання лабораторної роботи №2

Побудова піраміди. Пірамідальне сортування. Черга з пріоритетами.

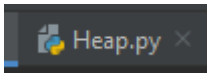
Виконав:
Студент групи ФЕП-22
Серафим Д.В.

Львів-2022

Хід роботи:

Частина 1. Побудова піраміди. Пірамідальне сортування.

1. Створити нову бібліотеку Heap (файли Heap.h, Heap.cpp).



2. У бібліотеці Heap, створити функції Parent(…), Left(…) та Right(…) для навігації по піраміді.

```
def left(i):
    return 2 * i + 1


def right(i):
    return 2 * i + 2
```

3. У бібліотеці Heap, згідно описаних в теоретичній частині алгоритмів, створити функції BuildMaxHeap(…) та MaxHeapify(…) для побудови та підтримки властивості незростаючої піраміди.

```
def max_heapify(array_inside, k):
    l = left(k)
    r = right(k)
    if l <= len(array_inside) - 1 and array_inside[l] > array_inside[k]:
        largest = l
    else:
        largest = k
    if r <= len(array_inside) - 1 and array_inside[r] > array_inside[largest]:
        largest = r
    if largest != k:
        array_inside[k], array_inside[largest] = array_inside[largest],
array_inside[k]
        max_heapify(array_inside, largest)


def build_min_heap(array_inside):
    n = int((len(array_inside) // 2) - 1)
    for k in range(n, -1, -1):
        min_heapify(array_inside, k)
```

4. У цій же бібліотеці створити функції BuildMinHeap(…) та MinHeapify(…). для побудови та підтримки властивості неспадної піраміди. Реалізація цих функцій є дзеркально симетричною до функцій BuildMaxHeap(…) та MaxHeapify(…).

```
def min_heapify(array_inside, k):
    l = left(k)
    r = right(k)
    if l < len(array_inside) and array_inside[l] < array_inside[k]:
        smallest = l
```

```
    else:
        smallest = k
    if r < len(array_inside) and array_inside[r] < array_inside[smallest]:
        smallest = r
    if smallest != k:
        array_inside[k], array_inside[smallest] = array_inside[smallest],
array_inside[k]
        min_heapify(array_inside, smallest)


def build_min_heap(array_inside):
    n = int((len(array_inside) // 2) - 1)
    for k in range(n, -1, -1):
        min_heapify(array_inside, k)
```

5. У бібліотеці Heap, згідно описаного в теоретичній частині алгоритму, створити функцію HeapSort(…) для реалізації пірамідального сортування одномірного масиву даних. У функції HeapSort(…) передбачити можливість сортування за зростанням (виклик функції BuildMaxHeap(…)) чи за спаданням (виклик функції BuildMinHeap(…)).

```
def heap_sort(array_inside, i):
    sort = []
    if i == 1:
        copy_a = array_inside.copy()
        for i in range(len(array_inside), 0, -1):
            build_max_heap(copy_a)
            sort.append(copy a[0])
            copy_a[0] = -100000
        return sort
    else:
        copy_a = array_inside.copy()
        for i in range(len(copy_a), 0, -1):
            build_min_heap(copy_a)
            sort.append(copy_a[0])
            copy_a[0] = 100000
        return sort
```

6. Створити новий проект Lab_3_1 та підключити до нього бібліотеку Heap. У функції main() проекту реалізувати меню для введення одномірного масиву даних, побудову на основі цих даних незростаючої чи неспадної піраміди, сортування введених даних за зростанням чи за спаданням, відображення результатів.

```
if not main_array:
    try:

        array_length = int(input("Введіть довжину масива:"))

        how_fill_array = int(input("Введіть яким чином хочете заповнити
```

```
масив:\n0 - Автоматично, з допомогою генератора.\n1 - Вручну, кожен
елемент.\n"))

        if how_fill_array == 0:

            print("Введіть границі генератора:")
            left_gen_cordon = int(input("Введіть яке найменше число може бути в
масиві:"))
            right_gen_cordon = int(input("Введіть яке найбільше число може бути
в масиві:"))

            for i in range(array_length):
                main_array.append(randint(left_gen_cordon, right_gen_cordon))

            program_menu()

        elif how_fill_array == 1:

            for x in range(array_length):
                num = int(input("Введіть число:"))
                main_array.append(num)

            program_menu()

    except ValueError:

        print("Щось пішло не так.\nСкоріше всього було введено неправильні вхідні
дані!")
```

7. Відкомпілювати проект та продемонструвати його роботу для одномірного масиву даних, отриманого від викладача.

```
Введіть довжину масива:20
Введіть яким чином хочете заповнити масив:
0 - Автоматично, з допомогою генератора.
1 - Вручну, кожен елемент.
0
Введіть границі генератора:
Введіть яке найменше число може бути в масиві:-10
Введіть яке найбільше число може бути в масиві:20
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
1

                    6
           5                       -6
     3           -2          -4          -9
  18    7     10    -3   -10    7     1     2
 13 0  9  4  15
--------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
```

```
6 - Заміна
3
Як сортуємо?
0 - за зростанням
1 - за спаданням
0
Відсортований масив:[-10, -9, -6, -4, -3, -2, 0, 1, 2, 3, 4, 5, 6, 7, 7, 9, 10,
13, 15, 18]
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
3
Як сортуємо?
0 - за зростанням
1 - за спаданням
1
Відсортований масив:[18, 15, 13, 10, 9, 7, 7, 6, 5, 4, 3, 2, 1, 0, -2, -3, -4, -
6, -9, -10]
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
2
Що будуємо?
0 - max_heap - Кучу де елементи зверху до низу з більшого до меншого
1 - min_heap - Кучу де елементи зверху до низу з меншого до більшого
0
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
1

                    18
            15                      7
     13          10          -4          2
  5     9     6     -3   -10    -6    1     -9
 3  0  7  4  -2
--------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
2
Що будуємо?
0 - max_heap - Кучу де елементи зверху до низу з більшого до меншого
1 - min_heap - Кучу де елементи зверху до низу з меншого до більшого
1
Введіть:
```

```
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
1

                    -10
          -3                      -9
     0          -2          -6          1
  3     4     6     10    -4    7     18    2
 13 5  7  9  15
--------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
```

Частина 2. Черга з пріоритетами.

1. У бібліотеці Heap, згідно описаних в теоретичній частині алгоритмів, створити функції HeapMax(…), HeapExtractMax(…), HeapIncreaseKey(…) та MaxHeapInsert(…) для роботи з чергою з пріоритетами, побудованою на основі незростаючої піраміди. Ці функції повинні, відповідно, повертати значення максимального елементу черги, видаляти максимальний елемент, змінювати (збільшувати) значення елементу, додавати елемент у чергу. У функції HeapIncreaseKey(…) передбачити обробку помилки зменшення ключа.

2. У цій же бібліотеці створити функції HeapMin(…), HeapExtractMin(…), HeapDecreaseKey(…) та MinHeapInsert(…) для роботи з чергою з пріоритетами, побудованою на основі неспадної піраміди. Реалізація цих функцій є дзеркально симетричною до функцій HeapMax(…), HeapExtractMax(…), HeapIncreaseKey(…) та MaxHeapInsert(…). У функції HeapDecreaseKey(…) передбачити обробку помилки збільшення ключа.

1 and 2

```
def heap_max(array):
    build_max_heap(array)
    return array[0]


def heap_extract_max(array):
    print(f"Число {heap_max(array,)} - максимальне,видаленно зі списку")
    array.pop(0)
    build_max_heap(array)
    heap_print(array)


def heap_increase_key(array_inside, i, key):

    if key < array_inside[i - 1]:  # Міняю значення на інше перебудовую дерево
        print("lol")
        # except "wrong!"
    array_inside[i - 1] = key
    while i > 0 and array_inside[(i // 2) - 1] < array_inside[i - 1]:
        array_inside[i - 1], array_inside[(i // 2) - 1] = array_inside[(i // 2)
- 1], array_inside[i - 1]
        i = (i) // 2
        heap_sort(array_inside, min_heapify)


def insert_max(array, new_num_for_add):
    build_max_heap(array)
    array.append(new_num_for_add)
    build_max_heap(array)
    heap_print(array)


def heap_min(array):
    build_min_heap(array)
    return array[0]


def heap_extract_min(array):
    print(f"Число {heap_min(array)} - мінімальне,видаленно зі списку")
    array.pop(0)
    build_min_heap(array)
    heap_print(array)


def replace_key(array_inside, i, key):
    array_inside[i] = key
    heap_print(array_inside)


def insert_min(array, new_num_for_add):
    build_min_heap(array)
    array.append(new_num_for_add)
    build_min_heap(array)
    heap_print(array)
```

3. Створити новий проект Lab_3_2 та підключити до нього бібліотеку Heap. У функції main() проекту реалізувати меню для введення одномірного масиву даних, побудову на основі цих даних

незростаючої чи неспадної піраміди та виконання описаних у п.1 та п.2 операцій для черги з пріоритетами, а також відображення результатів.

```
def program_menu():
    while True:
        try:

            x = int(input(
                "Введіть:\n0 - Закінчити роботу програми\n1 - Вивести\n2 - Будувати\n3 - Сортувати\n4 - Видалення і повернення елементу\n5 - Додавання елементу\n6 - Заміна\n"))

            if x == 0:
                break

            elif x == 1:

                heap_print(main_array)

            elif x == 2:

                print("Що будуємо?")
                y = int(input("0 - max_heap - Кучу де елементи зверху до низу з більшого до меншого\n1 - min_heap - Кучу де елементи зверху до низу з меншого до більшого\n"))

                if y == 0:
                    build_max_heap(main_array)
                elif y == 1:
                    build_min_heap(main_array)

            elif x == 3:

                print("Як сортуємо?")
                y = int(input("0 - за зростанням\n1 - за спаданням\n"))
                print(f"Відсортований масив:{heap_sort(main_array, y)}")

            elif x == 4:

                y = int(input("0 - max\n1 - min\n"))
                if y == 0:
                    heap_extract_max(main_array)
                else:
                    heap_extract_min(main_array)

            elif x == 5:

                y = int(input("0 - max\n1 - min\n"))
                number_for_add = int(input("Введіть число для додавання:"))
                if y == 0:
                    insert_max(main_array, number_for_add)
                else:
                    insert_min(main_array, number_for_add)

            elif x == 6:

                print(f"Масив:{main_array}")
                y = int(input("Введіть число яке в несеться в масив:"))
                number_for_add = int(input("Введіть індекс числа яке зміниться:"))
                replace_key(main_array, number_for_add, y)
```

```
        except ValueError:

            print("Щось пішло не так.\nСкоріше всього було введено неправильні
вхідні дані!\nПопробуйте ще раз!")

            continue


main_array = []

if not main_array:
    try:

        array_length = int(input("Введіть довжину масива:"))

        how_fill_array = int(input("Введіть яким чином хочете заповнити
масив:\n0 - Автоматично, з допомогою генератора.\n1 - Вручну, кожен
елемент.\n"))

        if how_fill_array == 0:

            print("Введіть границі генератора:")
            left_gen_cordon = int(input("Введіть яке найменше число може бути в
масиві:"))
            right_gen_cordon = int(input("Введіть яке найбільше число може бути
в масиві:"))

            for i in range(array_length):
                main_array.append(randint(left_gen_cordon, right_gen_cordon))

            program_menu()

        elif how_fill_array == 1:

            for x in range(array_length):
                num = int(input("Введіть число:"))
                main_array.append(num)

            program_menu()

    except ValueError:

        print("Щось пішло не так.\nСкоріше всього було введено неправильні вхідні
дані!")
```

4. Відкомпілювати проект та продемонструвати його роботу для одномірного масиву даних, отриманого від викладача.

```
Введіть довжину масива:20
Введіть яким чином хочете заповнити масив:
0 - Автоматично, з допомогою генератора.
1 - Вручну, кожен елемент.
0
Введіть границі генератора:
Введіть яке найменше число може бути в масиві:-10
Введіть яке найбільше число може бути в масиві:20
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
```

```
6 - Заміна
1

                        -9
            4                           13
     11           0            12           7
  10    -10     -1     0     15     4     -8     0
 11 -1 3  -9 9
--------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
4
0 - max
1 - min
0
Число 15 - максимальне,видаленно зі списку

                        13
            12                          11
     9            11           7            10
  3     0      0      -9     4     -8     0     4
 -1-10 -9 -1
--------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
4
0 - max
1 - min
1
Число -10 - мінімальне,видаленно зі списку

                        -9
            -9                          -1
     -8           4            0            3
  -1     0      11     11     7     10     4     9
 13 0  12
--------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
5
0 - max
1 - min
0
Введіть число для додавання:144

                       144
            13                          10
     12           11           7            9
```

```
  -1    0     4     11    -1    0     4     3
 -9 -8 -9 0
-------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
5
0 - max
1 - min
1
Введіть число для додавання:34

                        -9
            -9                        -1
     -8          4           0           3
  -1    0     11    11    7     10    4     9
144 12 13 0  34
-------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
6
Масив:[-9, -9, -1, -8, 4, 0, 3, -1, 0, 11, 11, 7, 10, 4, 9, 144, 12, 13, 0, 34]
Введіть число яке в несеться в масив:123
Введіть індекс числа яке зімінеться:4

                        -9
            -9                        -1
     -8         123          0           3
  -1    0     11    11    7     10    4     9
144 12 13 0  34
-------------------------------------------------
Введіть:
0 - Закінчити роботу програми
1 - Вивести
2 - Будувати
3 - Сортувати
4 - Видалення і повернення елементу
5 - Додавання елементу
6 - Заміна
```

Висновок: На цій лабораторній роботі я ознайомився з такими темами: Побудова піраміди. Пірамідальне сортування. Черга з пріоритетами.

Весь код:

```
"""IMPORT"""
from io import StringIO
import math
from random import randint


def heap_print(tree, total_width=50):  # Прінт дерева
    """
    :param tree:         Масив дерева
    :param total_width:  Ширина дерева ,за замовчуванням 50 символів
    :return:             Нічого
    """
    fill = " "
    output = StringIO()
    last_row = -1
    for i, n in enumerate(tree):
        if i:
            row = int(math.floor(math.log(i + 1, 2)))
        else:
            row = 0
        if row != last_row:
            output.write("\n")
        columns = 2**row
        col_width = int(math.floor((total_width * 1.0) / columns))
        output.write(str(n).center(col_width, fill))
        last_row = row
    print(output.getvalue())
    print("-" * total_width)
    return


def left(i):
    return 2 * i + 1


def right(i):
    return 2 * i + 2


def min_heapify(array_inside, k):
    l = left(k)
    r = right(k)
    if l < len(array_inside) and array_inside[l] < array_inside[k]:
        smallest = l
    else:
        smallest = k
    if r < len(array_inside) and array_inside[r] < array_inside[smallest]:
        smallest = r
    if smallest != k:
        array_inside[k], array_inside[smallest] = array_inside[smallest],
array_inside[k]
        min_heapify(array_inside, smallest)


def build_min_heap(array_inside):
    n = int((len(array_inside) // 2) - 1)
    for k in range(n, -1, -1):
        min_heapify(array_inside, k)


def max_heapify(array_inside, k):
    l = left(k)
    r = right(k)
    if l <= len(array_inside) - 1 and array_inside[l] > array_inside[k]:
```

```
        largest = l
    else:
        largest = k
    if r <= len(array_inside) - 1 and array_inside[r] > array_inside[largest]:
        largest = r
    if largest != k:
        array_inside[k], array_inside[largest] = array_inside[largest],
array_inside[k]
        max_heapify(array_inside, largest)


def build_max_heap(array_inside):
    n = len(array_inside) // 2 - 1
    for k in range(n, -1, -1):
        max_heapify(array_inside, k)


def heap_sort(array_inside, i):
    sort = []
    if i == 1:
        copy_a = array_inside.copy()
        for i in range(len(array_inside), 0, -1):
            build_max_heap(copy_a)
            sort.append(copy_a[0])
            copy_a[0] = -100000
        return sort
    else:
        copy_a = array_inside.copy()
        for i in range(len(copy_a), 0, -1):
            build_min_heap(copy_a)
            sort.append(copy_a[0])
            copy_a[0] = 100000
        return sort


def heap_max(array):
    build_max_heap(array)
    return array[0]


def heap_extract_max(array):
    print(f"Число {heap_max(array,)} - максимальне,видаленно зі списку")
    array.pop(0)
    build_max_heap(array)
    heap_print(array)


def heap_increase_key(array_inside, i, key):

    if key < array_inside[i - 1]:  # Міняю значення на інше перебудовую дерево
        print("lol")
        # except "wrong!"
    array_inside[i - 1] = key
    while i > 0 and array_inside[(i // 2) - 1] < array_inside[i - 1]:
        array_inside[i - 1], array_inside[(i // 2) - 1] = array_inside[(i // 2)
- 1], array_inside[i - 1]
        i = (i) // 2
        heap_sort(array_inside, min_heapify)


def insert_max(array, new_num_for_add):
    build_max_heap(array)
    array.append(new_num_for_add)
    build_max_heap(array)
    heap_print(array)
```

```
def heap_min(array):
    build_min_heap(array)
    return array[0]


def heap_extract_min(array):
    print(f"Число {heap_min(array)} - мінімальне,видаленно зі списку")
    array.pop(0)
    build_min_heap(array)
    heap_print(array)


def replace_key(array_inside, i, key):
    array_inside[i] = key
    heap_print(array_inside)


def insert_min(array, new_num_for_add):
    build_min_heap(array)
    array.append(new_num_for_add)
    build_min_heap(array)
    heap_print(array)


def program_menu():
    while True:
        try:

            x = int(input(
                "Введіть:\n0 - Закінчити роботу програми\n1 - Вивести\n2 -
Будувати\n3 - Сортувати\n4 - Видалення і повернення елементу\n5 - Додавання
елементу\n6 - Заміна\n"))

            if x == 0:
                break

            elif x == 1:

                heap_print(main_array)

            elif x == 2:

                print("Що будуємо?")
                y = int(input("0 - max_heap - Кучу де елементи зверху до низу з
більшого до меншого\n1 - min_heap - Кучу де елементи зверху до низу з меншого до
більшого\n"))

                if y == 0:
                    build_max_heap(main_array)
                elif y == 1:
                    build_min_heap(main_array)

            elif x == 3:

                print("Як сортуємо?")
                y = int(input("0 - за зростанням\n1 - за спаданням\n"))
                print(f"Відсортований масив:{heap_sort(main_array, y)}")

            elif x == 4:

                y = int(input("0 - max\n1 - min\n"))
                if y == 0:
                    heap_extract_max(main_array)
                else:
                    heap_extract_min(main_array)
```

```
            elif x == 5:

                y = int(input("0 - max\n1 - min\n"))
                number_for_add = int(input("Введіть число для додавання:"))
                if y == 0:
                    insert_max(main_array, number_for_add)
                else:
                    insert_min(main_array, number_for_add)

            elif x == 6:

                print(f"Масив:{main_array}")
                y = int(input("Введіть число яке в несеться в масив:"))
                number_for_add = int(input("Введіть індекс числа яке
зімінеться:"))
                replace_key(main_array, number_for_add, y)

        except ValueError:

            print("Щось пішло не так.\nСкоріше всього було введено неправильні
вхідні дані!\nПопробуйте ще раз!")

            continue


main_array = []

if not main_array:
    try:

        array_length = int(input("Введіть довжину масива:"))

        how_fill_array = int(input("Введіть яким чином хочете заповнити
масив:\n0 - Автоматично, з допомогою генератора.\n1 - Вручну, кожен
елемент.\n"))

        if how_fill_array == 0:

            print("Введіть границі генератора:")
            left_gen_cordon = int(input("Введіть яке найменше число може бути в
масиві:"))
            right_gen_cordon = int(input("Введіть яке найбільше число може бути
в масиві:"))

            for i in range(array_length):
                main_array.append(randint(left_gen_cordon, right_gen_cordon))

            program_menu()

        elif how_fill_array == 1:

            for x in range(array_length):
                num = int(input("Введіть число:"))
                main_array.append(num)

            program_menu()

    except ValueError:

        print("Щось пішло не так.\nСкоріше всього було введено неправильні вхідні
дані!")
```