CS102L Data Structures & Algorithms
Lab 5

Department of Computer Science

Dhanani School of Science and Engineering

Habib University

Spring 2024

Contents

Introduction

1.1 Restriction

For this lab, you are again working on fixed size arrays and hence, are not allowed to use append, remove, pop, or any similar function that changes the size of a fixed size array for this lab.
If you want to initialize a new array, use your codes from previous labs for this purpose. Additionally, code for initializing arrays in python is also added in pre-requisites section in this PDF.

In addition to this, python builtin functions such as find, sort, in or any similar functions are not allowed either. Only function which is allowed to use is len.

1.2 Instructions

- This lab will contribute 1% towards your final grade.
- The deadline for submission of this lab is at the end of lab time.
- The lab must be submitted online via CANVAS. You are required to submit a zip file that contains all the *.py* files.
- The zip file should be named as *Lab_05_aa1234.zip* where *aa1234* will be replaced with your student id.
- **Files that don't follow the appropriate naming convention will not be graded.**
- **Your final grade will comprise of both your submission and your lab performance.**

1.3 Marking scheme

This lab will be marked out of 100.

- 50 Marks are for the completion of the lab.
- 50 Marks are for progress and attendance during the lab.

1.4 Lab Objectives

- To understand and implement different sorting algorithms in Python
- To use sorting algorithms in different exercises

1.5 Late submission policy

There is no late submission policy for the lab.

1.6 Use of AI

Taking help from any AI-based tools such as ChatGPT is strictly prohibited and will be considered plagiarism.

1.7 Viva

Course staff may call any student for Viva to provide an explanation for their submission.

Prerequisites

2.1 Array Initialization in python

Arrays are not built-in data structure inside python. However, lists can be simulated as arrays.
For example:
Initializing 1D Arrays in Python: To initialize a 1D array of 5 elements, we can do this:

```
arr = [0 for i in range(5)]
```

This initialized an array with 0's. Similarly, to initialize 2D arrays, we can do this:

```
arr = [[0 for i in range(5)] for j in range(5)]
```

Please visit any of the following links to understand how to initialize 2D arrays the right way:

- https://en.wikibooks.org/wiki/Python_Programming/Lists#List_creation_shortcuts
- https://www.geeksforgeeks.org/python-using-2d-arrays-lists-the-right-way/

2.2 Assert Statements

Assert statements are crucial tools in programming used to verify assumptions and test conditions within code. For instance, in Python, an assert statement is typically as follows:

```
    assert condition, message
```

For example, in a function calculating the square root, in order to ensure that the input value is non-negative, one might add the following assert statement:

```
assert x >= 0, "x should be a non-negative number"
```

If the condition evaluates to False, an AssertionError is raised, highlighting the issue and its corresponding message.

2.3 Pytest

Pytest is a popular testing framework in Python known for its simplicity and power in facilitating efficient and scalable testing for applications and software. In order to use Pytest, one first needs to install it and create test functions prefixed with *test_* in a Python file. To install Pytest, you can write the following command on the terminal:

```
pip install pytest
```

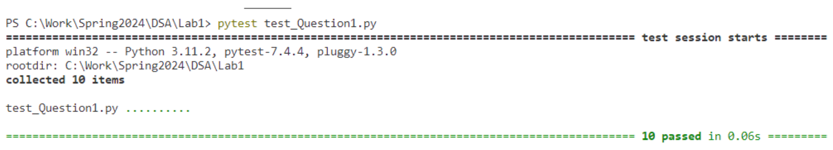
After installing Pytest, one can execute all the tests by typing the following command on the terminal:

```
pytest
```

If you want to test any particular file, then you will have to specify the filename as follows:

```
pytest <filename>.py
```

If all the tests pass successfully, then you should receive a similar output.

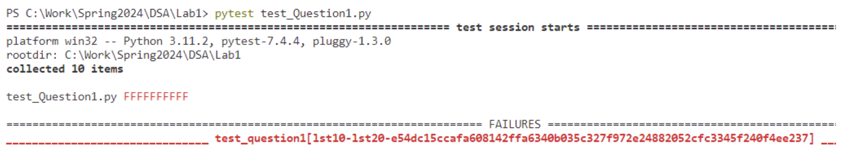


```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
============================================================================================ test session starts ========
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

test_Question1.py ..........

============================================================================================ 10 passed in 0.06s ========
```

However, if all the tests fail, then you should receive a similar output.



```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
==================================================================== test session starts ======================================
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

test_Question1.py FFFFFFFFFF

======================================================================== FAILURES ============================================
______________________________ test_question1[lst10-lst20-e54dc15ccafa608142ffa6340b035c327f972e24882052cfc3345f240f4ee237] __
```

Lab Exercises

3.1 Selection Sort

It is one of the simplest sorting algorithm. We repeatedly find the next largest (or smallest) element in the array and move it to its final position in the sorted array.

Selection Sort simulation is available here.

Algorithm

```
    repeat (numOfElements - 1) times
        set the first unsorted element as the minimum

        for each of the unsorted elements
            if element < currentMinimum
                set element as new minimum

        swap minimum with first unsorted position
```

3.1.1 Function Description

Write a function `selection_sort(arr)` that take an array and sort it using `Selection Sort` Algorithm. After each iteration, print the current state of the array.

This function should be implemented in the file **Question1.py**, accessible within the Lab 05 module on CANVAS.

3.1.2 Sample

```
>>> selection_sort([54, 26, 93, 17, 77, 31, 44, 55, 20])
[17, 26, 93, 54, 77, 31, 44, 55, 20]
[17, 20, 93, 54, 77, 31, 44, 55, 26]
[17, 20, 26, 54, 77, 31, 44, 55, 93]
[17, 20, 26, 31, 77, 54, 44, 55, 93]
[17, 20, 26, 31, 44, 54, 77, 55, 93]
```

```
[17, 20, 26, 31, 44, 54, 77, 55, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]

>>> selection_sort(["Aisha", "Nadia", "Waqar", "Saleha", "
   Hasan", "Shahid", "Shah Jamal", "Abdullah", "Umair", "Taj
   "])
['Abdullah', 'Nadia', 'Waqar', 'Saleha', 'Hasan', 'Shahid',
   'Shah Jamal', 'Aisha', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Waqar', 'Saleha', 'Hasan', 'Shahid',
   'Shah Jamal', 'Nadia', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Saleha', 'Waqar', 'Shahid',
   'Shah Jamal', 'Nadia', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Waqar', 'Shahid', '
   Shah Jamal', 'Saleha', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shahid',
   'Shah Jamal', 'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
```

3.1.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question1.py
```

3.2 Insertion Sort

Insertion sort is a simple sorting algorithm, a comparison sort in which the sorted array (or list) is built one entry at a time.

Insertion Sort simulation is available here.

Algorithm

```
mark first element as sorted
for each unsorted element X
    'extract' the element X
    for j = lastSortedIndex down to 0
        if current element j > X
            move sorted element to the right by 1
    break loop and insert X here
```

3.2.1 Function Description

Write a function insertion_sort(arr) that take a array and sort it using Insertion Sort Algorithm. Print state after each iteration just like previous question.

This function should be implemented in the file **Question2.py**, accessible within the Lab 05 module on CANVAS.

3.2.2 Sample

```
>>> insertion_sort([54, 26, 93, 17, 77, 31, 44, 55, 20])
[26, 54, 93, 17, 77, 31, 44, 55, 20]
[26, 54, 93, 17, 77, 31, 44, 55, 20]
[17, 26, 54, 93, 77, 31, 44, 55, 20]
[17, 26, 54, 77, 93, 31, 44, 55, 20]
[17, 26, 31, 54, 77, 93, 44, 55, 20]
[17, 26, 31, 44, 54, 77, 93, 55, 20]
[17, 26, 31, 44, 54, 55, 77, 93, 20]
[17, 20, 26, 31, 44, 54, 55, 77, 93]

>>> insertion_sort(["Aisha", "Nadia", "Waqar", "Saleha", "
    Hasan", "Shahid", "Shah Jamal", "Abdullah", "Umair", "Taj
    "])
['Aisha', 'Nadia', 'Waqar', 'Saleha', 'Hasan', 'Shahid', '
    Shah Jamal', 'Abdullah', 'Umair', 'Taj']
['Aisha', 'Nadia', 'Waqar', 'Saleha', 'Hasan', 'Shahid', '
    Shah Jamal', 'Abdullah', 'Umair', 'Taj']
['Aisha', 'Nadia', 'Saleha', 'Waqar', 'Hasan', 'Shahid', '
    Shah Jamal', 'Abdullah', 'Umair', 'Taj']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Waqar', 'Shahid', '
    Shah Jamal', 'Abdullah', 'Umair', 'Taj']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shahid', 'Waqar', '
    Shah Jamal', 'Abdullah', 'Umair', 'Taj']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid'
    , 'Waqar', 'Abdullah', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
    Jamal', 'Shahid', 'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
    Jamal', 'Shahid', 'Umair', 'Waqar', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
    Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
```

3.2.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question2.py
```

3.3 Bubble Sort

The bubble sort makes multiple passes through an array (or list). It compares adjacent items and exchanges those that are out of order. Each pass through the array places the next largest value in its proper place. In essence, each item "bubbles" up to the location where it belongs.

Bubble Sort simulation is available here.

Algorithm

```
for x = 0 to n
    for i = 0 to indexOfLastUnsortedElement-1
        if leftElement > rightElement
            swap(leftElement, rightElement)
```

3.3.1 Function Description

Write a function `bubble_sort(arr)` that take a array and sort it using `Bubble Sort Algorithm`. Print state after each iteration just like previous question.

This function should be implemented in the file **Question3.py**, accessible within the Lab 05 module on CANVAS.

3.3.2 Sample

```
>>> bubble_sort([54, 26, 93, 17, 77, 31, 44, 55, 20])
[26, 54, 17, 77, 31, 44, 55, 20, 93]
[26, 17, 54, 31, 44, 55, 20, 77, 93]
[17, 26, 31, 44, 54, 20, 55, 77, 93]
[17, 26, 31, 44, 20, 54, 55, 77, 93]
[17, 26, 31, 20, 44, 54, 55, 77, 93]
[17, 26, 20, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]

>>> bubble_sort(["Aisha", "Nadia", "Waqar", "Saleha", "Hasan
   ", "Shahid", "Shah Jamal", "Abdullah", "Umair", "Taj"])
['Aisha', 'Nadia', 'Saleha', 'Hasan', 'Shahid', 'Shah Jamal'
   , 'Abdullah', 'Umair', 'Taj', 'Waqar']
```

```
['Aisha', 'Nadia', 'Hasan', 'Saleha', 'Shah Jamal', '
   Abdullah', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Abdullah', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Aisha', 'Hasan', 'Nadia', 'Abdullah', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Aisha', 'Hasan', 'Abdullah', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Aisha', 'Abdullah', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah
   Jamal', 'Shahid', 'Taj', 'Umair', 'Waqar']
```

3.3.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question3.py
```

3.4 Sort Matrix By Column Number

3.4.1 Function Description

Using `Selection Sort`, Write a function `sort_matrix_by_columnNumber` which takes as parameter a matrix and a columnNumber and sort it *by column number*. You need to return the sorted matrix.

This function should be implemented in the file **Question4.py**, accessible within the Lab 05 module on CANVAS.

3.4.2 Sample

```
>>> sort_matrix_by_columnNumber([[5, 8, 1], [6, 7, 3], [5,
   4, 9]], 0)
[[5, 8, 1], [5, 4, 9], [6, 7, 3]]

>>> sort_matrix_by_columnNumber([['square', 'rectangle', '
   triangle'],['chair','table', 'house'],['motor cycle', '
   car', 'truck']], 2)
[['chair', 'table', 'house'], ['square', 'rectangle', '
   triangle'], ['motor cycle', 'car', 'truck']]
```

3.4.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question4.py
```

3.5 Sorting Rectangle Records

Suppose that we have the following rectangle records in an array (or list) of dictionaries:

```
rectangle_records = [{"ID": "Rect1", "Length": 40, "Breadth"
   : 25, "Color": "red"}, {"ID": "Rect2", "Length": 30, "
   Breadth": 20, "Color": "blue"}, {"ID": "Rect3", "Length":
    70, "Breadth": 45, "Color": "green"}, {"ID": "Rect4", "
   Length": 20, "Breadth": 10, "Color": "purple"}]
```

You are required to sort the rectangle_records in *ascending order* by record_title. The value of record_title will be *ID*, *Length*, *Breadth*, or *Color*.

3.5.1 Function Description

Using `Insertion Sort`, Write a function `sort_rectangles` that takes rectangle_records and record_title as a parameter and sort the rectangle_records in *ascending order* by record_title. You need to return the updated rectangle_records.

This function should be implemented in the file **Question5.py**, accessible within the Lab 05 module on CANVAS.

3.5.2 Sample

```
>> rectangle_records = [{"ID": "Rect1", "Length": 40, "
   Breadth": 25, "Color": "red"}, {"ID": "Rect2", "Length":
   30, "Breadth": 20, "Color": "blue"}, {"ID": "Rect3", "
   Length": 70, "Breadth": 45, "Color": "green"}, {"ID": "
   Rect4", "Length": 20, "Breadth": 10, "Color": "purple"}]

>>> sort_rectangles(rectangle_records, "ID")
[{"ID": "Rect1", "Length": 40, "Breadth": 25, "Color": "red"
   }, {"ID": "Rect2", "Length": 30, "Breadth": 20, "Color":
   "blue"}, {"ID": "Rect3", "Length": 70, "Breadth": 45, "
   Color": "green"}, {"ID": "Rect4", "Length": 20, "Breadth"
   : 10, "Color": "purple"}]

>>> sort_rectangles(rectangle_records, "Length")
[{'ID': 'Rect4', 'Length': 20, 'Breadth': 10, 'Color': '
   purple'}, {'ID': 'Rect2', 'Length': 30, 'Breadth': 20, '
   Color': 'blue'}, {'ID': 'Rect1', 'Length': 40, 'Breadth':
    25, 'Color': 'red'}, {'ID': 'Rect3', 'Length': 70, '
   Breadth': 45, 'Color': 'green'}]

>>> sort_rectangles(rectangle_records, "Breadth")
[{'ID': 'Rect4', 'Length': 20, 'Breadth': 10, 'Color': '
   purple'}, {'ID': 'Rect2', 'Length': 30, 'Breadth': 20, '
   Color': 'blue'}, {'ID': 'Rect1', 'Length': 40, 'Breadth':
    25, 'Color': 'red'}, {'ID': 'Rect3', 'Length': 70, '
   Breadth': 45, 'Color': 'green'}]

>>> sort_rectangles(rectangle_records, "Color")
[{'ID': 'Rect2', 'Length': 30, 'Breadth': 20, 'Color': 'blue
   '}, {'ID': 'Rect3', 'Length': 70, 'Breadth': 45, 'Color':
    'green'}, {'ID': 'Rect4', 'Length': 20, 'Breadth': 10, '
   Color': 'purple'}, {'ID': 'Rect1', 'Length': 40, 'Breadth
   ': 25, 'Color': 'red'}]
```

3.5.3 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question5.py
```