CS102L Data Structures & Algorithms
Lab week 8

Department of Computer Science

Dhanani School of Science and Engineering

Habib University

Spring 2024

Contents

Introduction

1.1 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_07_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.**

1.2 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.

1.3 Lab Objectives

- To understand and implement quick sort, using different pivot selection schemes, in Python.
- To use quick sort in different exercises.

1.4 Late submission policy

There is no late submission policy for the lab.

1.5 Use of AI

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

1.6 Viva

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

Prerequisites

2.1 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:

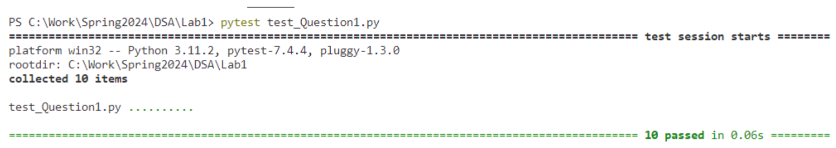
```
pytest <filename>.py
```

In case the above mentioned pytest commands do not work on your system, even though pytest is installed, try using the following commands:

```
py -m pytest
```

```
py -m pytest <filename>.py
```

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

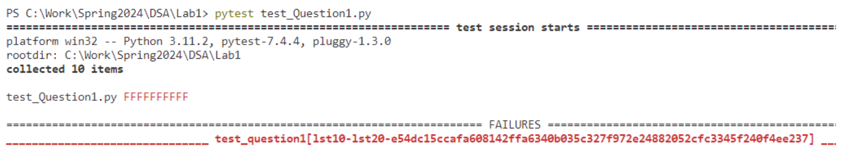


```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
============================================================================================ test session starts ========
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

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] __
```

Exercises

3.1 QuickSort Algorithm

Write a function named `quick_sort`, which takes the following three inputs:

- lst – An input list of unsorted data.
- low – An integer denoting the lowest index of the list.
- high - An integer denoting the highest index of the list.

The following psuedocodes provide a simple implementation for the QuickSort Algorithm:

Algorithm 1 Partition Function

```
Require: $lst$: list to partition, $low$: starting index, $high$: ending index
Ensure: index of the pivot element
 1: $pivot \leftarrow low$
 2: $i \leftarrow low + 1$
 3: for $j$ from $low + 1$ to $high + 1$ do
 4:     if $lst[j] \leq lst[pivot]$ then
 5:         swap $lst[i]$, $lst[j]$
 6:         $i \leftarrow i + 1$
 7:     end if
 8: end for
 9: swap $lst[pivot]$, $lst[i - 1]$
10: $pivot \leftarrow i - 1$
11: return $pivot$
```

Algorithm 2 PartitionMid Function

```
Require: $lst$: list to partition, $low$: starting index, $high$: ending index
Ensure: index of the pivot element
 1: $pivot \leftarrow$ middle index of lst
 2: swap $lst[low]$, $lst[pivot]$
 3: return Partition($lst, low, high$)
```

Algorithm 3 QuickSort Function

```
Require: $lst$: list to sort, $low$: starting index, $high$: ending index
1: if $low < high$ then
2:     $pi \leftarrow$ PartitionMid$(lst, low, high)$
3:     print$(lst)$
4:     QuickSort$(lst, low, pi - 1)$
5:     QuickSort$(lst, pi + 1, high)$
6: end if
```

The function should use the middle element at the pivot and sort the given list inplace, without returning the array. It should print the array right after the partition helper function is called and before the recursive calls of the function.

3.1.1 Examples

```
>>> lst = [5, 9, 2, 1, 6, 3, 0]
>>> quick_sort(lst, 0, len(lst) - 1)
[0, 1, 2, 5, 6, 3, 9]
[0, 1, 3, 5, 2, 6, 9]
[0, 1, 2, 3, 5, 6, 9]
[0, 1, 2, 3, 5, 6, 9]

>>> lst = [21, 36, 11, 9, 6, 42, 39]
>>> quick_sort(lst, 0, len(lst) - 1)
[6, 9, 11, 21, 36, 42, 39]
[6, 9, 11, 21, 36, 42, 39]
[6, 9, 11, 21, 36, 42, 39]
[6, 9, 11, 21, 36, 39, 42]
```

3.1.2 Testing

To test your function, type one of the following commands in the terminal:

```
pytest test_Question1.py
```

```
py -m pytest test_Question1.py
```

3.2 Sort Matrix By Column Number

3.2.1 Function Description

Modify your implementation of `Quick Sort` from Q1 such that it uses the element with the highest index as the pivot. Using this updated algorithm, write a function `quick_sort_by_columnNumber` which takes as parameter a matrix, lowest index, highest index, and columnNumber and sort it *by column number*. Since

Quick Sort does the sorting inplace, you do not need to return the array.

This function should be implemented in the file **Question2.py**, accessible within the Lab 07 module on CANVAS.

3.2.2 Sample

```
>>> matrix = [[7, 8, 9], [1, 2, 3], [4, 5, 6]]
>>> quick_sort_by_column_number(matrix, 0, 2, 2)
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

>>> matrix = [['square', 'rectangle', 'triangle'],['chair',
   'table', 'house'],['motor cycle', 'car', 'truck']]
>>> quick_sort_by_column_number(matrix, 0, 2, 2)
[['square', 'rectangle', 'triangle'], ['chair', 'table', '
   house'], ['motor cycle', 'car', 'truck']]
[['chair', 'table', 'house'], ['square', 'rectangle', '
   triangle'], ['motor cycle', 'car', 'truck']]
```

3.2.3 Testing

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

```
pytest test_Question2.py
```

```
py -m pytest test_Question2.py
```

3.3 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*.

3.3.1 Function Description

Modify your implementation of Quick Sort from Q1, such that it uses the element with the lowest index as the pivot. Using this updated algorithm, write a function quick_sort_rectangles that takes rectangle_records, low, high, and record_title as parameters and sort the rectangle_records in *ascending order* by record_title.

Since `Quick Sort` does the sorting inplace, you do not need to return the array.

This function should be implemented in the file **Question3.py**, accessible within the Lab 07 module on CANVAS.

3.3.2 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"}]

>>> quick_sort_rectangles(rectangle_records, 0, len(
   rectangle_records)-1, "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'}]
[{'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'}]
```

3.3.3 Testing

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

```
pytest test_Question3.py
```

```
py -m pytest test_Question3.py
```