BLG 335E – Analysis of Algorithms I

Homework 1

Due by November 13, 2022 at 23:59

Important Notices:

- All codes and reports will be checked using a plagiarism detection tool and the similarities will be calculated.
- The plagiarism-detected submissions will be penalized with minus points and ITU discipline regulations apply to the authors of these submissions.
- Attention:
  - Please write the codes and reports on your own.
  - Make sure to cite any sources that you get help from.(You can look at the sources to understand the logic of the algorithms, provided that you do not copy the codes.)
  - Copying code parts from books, websites, or from a friend’s code is considered plagiarism.
  - Changing variable names and line orders in code that you get from a source does not mean you are not plagiarizing. The plagiarism detection tool can even detect such similarities.
  - Do not post your codes on any public platform (e.g. Github, DockerHub) before the deadline for homework.
- You may use STL but do not use built-in sorting functions.
- Do not forget to comment on your code to make the evaluation easier for us.

Program output:

- Print out the array before and after sorting, the execution time of the algorithm, and number of partitionings (iterations) during sorting.

Running the codes:

- All of you are expected to have installed Docker and made the necessary configurations by following the installation steps shared with you on Ninova.
- We are going to run all the codes using Docker. Therefore, make sure that your codes run on Docker without errors.
- It is compulsory to include a "How2Run" section in your report. This section gives all the details to compile and run your code.

Submission:

- Submit your source code files (upload all necessary files to run the code except the “books.csv” dataset) and report file in the format of “StudentNumber_HW1_Report.pdf” on Ninova before the deadline, late submissions and submissions via e-mail will not be accepted.

Good Luck! ☺

Part1. Implementation (48 points):

A dataset including book ratings has been uploaded to Ninova under HW-1 folder (“books.csv”).

The dataset has 12 attributes and 11K instances. You can learn more about the dataset (https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks). Yet, use the version uploaded to Ninova as there are a few corrections to the dataset.

One of the attributes is “average_rating”. In this homework, you are asked to sort this dataset with respect to the books’ average rating values and save it again as a .csv or .txt file.

Code the Quicksort algorithms with pseudocodes given below and sort the average rating values in the dataset. An example scenario is given for you to better understand how the algorithm works.

A. Quicksort-I (QS1)

Pseudocode-I:

```
An array “Arr” is given:
Arr = [....]

Initial configuration:
        Low = the first index of the Arr
        High = the last index of the Arr
        i , j = Low
        Pivot = Arr[High]
```

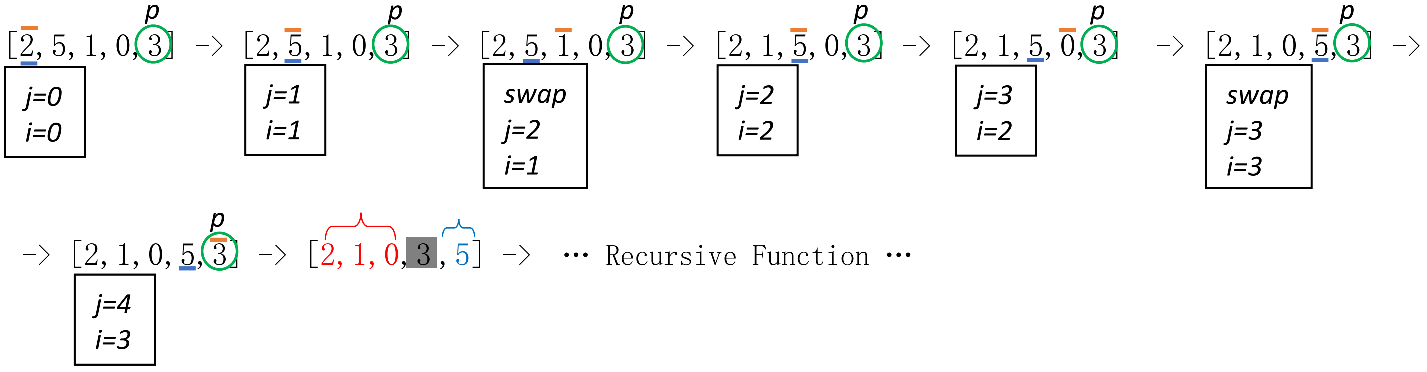
```
Under the condition j >= i    // j moves first
    While Arr[i]  <= Pivot and Arr[j] <= Pivot

        j++, i++
        No Swap

    While Arr[j] > Pivot

        j++
        Swap Arr[i] and Arr[j]

    After Swap update i. (check arr[i] <= Pivot)

    if  j == High then:
        Swap Arr[i] and Arr[High]

        break and return i   //for partitioning

         //1) From Low to i-1 , 2) From i+1 to High

REPEAT SORTING for these sub-arrays
```

Sample scenario-I:

Arr = [2, 5, 1, 0, 3]

i, *j* = 0

Low = 0, *High* = 4



B. Quicksort-II (QS2)

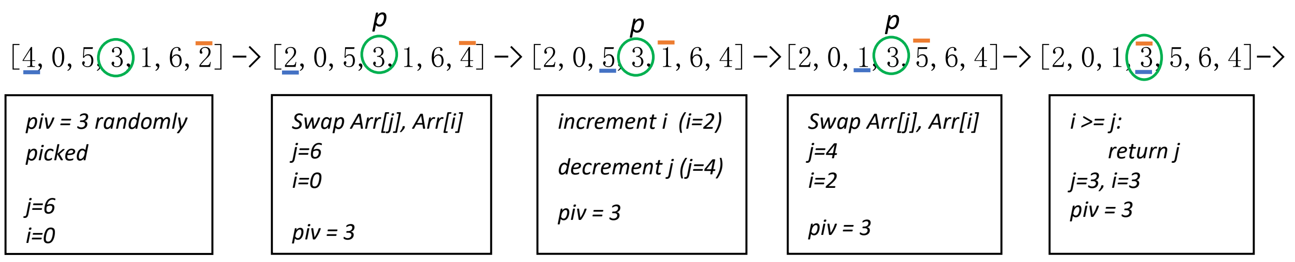
Pseudocode-II:

```
An array "Arr" is given:
Arr = [….]

Initial configuration:
        Low = the first index of the Arr
        High = the last index of the Arr
        i = Low
        j = High
        Pivot = will be randomly chosen
```

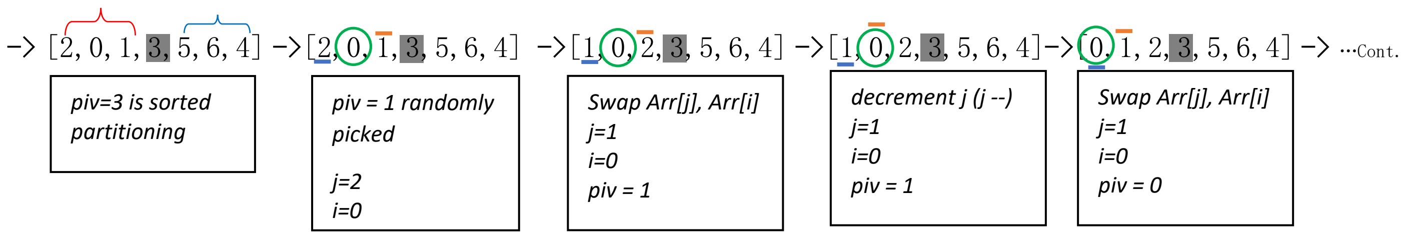
```
CHOOSE a RANDOM index "Rand" amoung Arr indices
Pivot = Rand
Under the condition j > i
    While Arr[i]<= Arr[Pivot]
        i++
    While Arr[j] >= Arr[Pivot]
        j- -
    Otherwise:
        SWAP Arr[i] and Arr[j]
    if  i >= j
        break and return j    // for partitioning two arrays
                // 1) From Low to j-1 , 2) From j+1 to High
REPEAT SORTING for these sub-arrays
NOTE THAT  Arr[j] >= Arr[Pivot] >= Arr[i]
```

Sample scenario-II:

```
Arr = [4, 0, 5, 3, 1, 6, 2]
i = 0
j = 6
```



p
p
p
[4, 0, 5, 3, 1, 6, 2] -> [2, 0, 5, 3, 1, 6, 4] -> [2, 0, 5, 3, 1, 6, 4] ->[2, 0, 1, 3, 5, 6, 4]-> [2, 0, 1, 3, 5, 6, 4]->
piv = 3 randomly picked j=6 i=0
Swap Arr[j], Arr[i] j=6 i=0 piv = 3
increment i (i=2) decrement j (j=4) piv = 3
Swap Arr[j], Arr[i] j=4 i=2 piv = 3
i >= j: return j j=3, i=3 piv = 3



-> [2, 0, 1, 3, 5, 6, 4] ->[2, 0, 1, 3, 5, 6, 4] ->[1, 0, 2, 3, 5, 6, 4] ->[1, 0, 2, 3, 5, 6, 4]->[0, 1, 2, 3, 5, 6, 4] -> …Cont.
piv=3 is sorted partitioning
piv = 1 randomly picked j=2 i=0
Swap Arr[j], Arr[i] j=1 i=0 piv = 1
decrement j (j --) j=1 i=0 piv = 1
Swap Arr[j], Arr[i] j=1 i=0 piv = 0

Part2. Report (**52 points**)

➢ Write the report *neatly* and use concise sentences that give sufficient answers rather than using too many sentences.

a. (**6 points**)As you asked in the implementation part write the codes for the given quicksort algorithms and add a “How2Run” section in your report.

b. (**6 points**) Read data from the books.csv and sort “average_rating values” in ascending order, and then rewrite the sorted results in the sorted_books.csv (or sorted_books.txt) file. You can use either QS1 or QS2 for this purpose.

c. (**10 points**) Calculate the time elapsed for each algorithm (QS1 & QS2) in the code. Print out some statistics regarding the algorithm such as the number of swaps, partitions, etc. Give screenshots of the outputs. Use half/quarter of the data and redo the calculations. Have there been any significant changes in the execution time?

d. (**12 points**) Which algorithm do you think is more efficient, Why? Write down the asymptotic upper bounds for the algorithms for the best case,the worst case and the average case. Prove them solving the recurrence equations (You may explain the details on your code/pseudecode.).

e. (**12 points**) Supply the worst case secenarios for QS1 & QS2 (schematic representation is preferred.). What do you think could be the reason for this? What method would you suggest to reduce time complexity in these scenarios? Explain.

f. (**6 points**) What is the effect of pivot randomization implemented in QS2? What purpose does it contribute exactly?