

# COSC 2P03 Advanced Data Structures: Assignment 3

Instructor: Yifeng Li<sup>\*1</sup>

<sup>1</sup>Department of Computer Science, Brock University

November 9, 2021

## 1 Introduction to the Problem

A drug is a small molecule that can bind to a target to inhibit or activate a pathway. However, not every compound can be used a medicine. To select drug candidates, a few physicochemical properties need to be first considered. In this assignment, you need to use the three sort algorithms - heap sort, quick sort, and merge sort, to sort molecules based on given properties.

## 2 Your Tasks (12 Marks)

1. Define a class named **DrugDesign** which should include data attributes named **originalData** and **finalSortedData**, and the following methods. Feel free to define other variable and method attributes in the class.
2. Define a method named **loadData** in the **DrugDesign** class. This method should take a filename (of course with absolute or relative path) as the input parameter and store the loaded data in the **originalData** variable. (A CSV file named **ZINCSubset.csv** is provided. You should load this file using method **loadData** in the main function). The **originalData** variable is an array with each element stores a row in the text file. Thus, each element of your array should be an object of a data item class with the following variables: **smiles** (string), **fragments** (string), **n\_fragments** (integer), **C** (integer), **F** (integer), **N** (integer), **O** (integer), **Other** (integer), **SINGLE** (integer), **DOUBLE** (integer), **TRIPLE** (integer), **Tri** (integer), **Quad** (integer), **Pent** (integer), **Hex** (integer), **logP** (float), **mr** (float), **qed** (float), and **SAS** (float) where corresponding data types are indicated in brackets. (2 marks)
3. Define a method named **heapSort** in the **DrugDesign** class to implement the max-heap based heap sort algorithm (in incremental order). This method should take a 1-D array as input, and finally return the sorted indices only as an array of integers. For example if you have an array with values [3,1,5,2,2,3,1,4,6,5], the method should return [ 3, 2, 0, 1, 4 ]. (2 marks)
4. Define a method named **quickSort** in the **DrugDesign** class to implement the quick-sort algorithm (in incremental order) discussed in our lectures. Similar to the above method, **quickSort** should take a 1-D array as input, and only return the sorted indices. (2 marks)
5. Define a method named **mergeSort** in the **DrugDesign** class to implement the merge sort algorithm (in incremental order) learned from this course. Similar to the above two methods, **mergeSort** should take a 1-D array as input, and only return the sorted indices. (2 marks)
6. Define a method named **sumRankSort** in the **DrugDesign** class. This method should take the three 1-D arrays of indices as input, take element-wise sum and store it in a local 1-D array named **sumOfRanks**, and then sort **sumOfRanks** using a method of your choice (**heapSort**, **quickSort**, or **mergeSort**), and finally return the sorted indices. (1 mark)
7. In the **main** function, you should following the following steps:

---

<sup>\*</sup>E-mail address: yli2@brocku.ca yifeng.li.cn@gmail.com

- (a) Create an instance of the `DrugDesign` class. (0.25 mark)
- (b) Call the `loadData` method to read data from `ZINCSubset.csv` to variable `originalData`. (0.25 mark)
- (c) Call the `heapSort` method to sort the original data based on property `logP`. Note that you should not really change `originalData`. Just obtain the sorted indices. (0.25 mark)
- (d) Call the `quickSort` method to sort the original data based on property `mr`. Note that you should not really change `originalData`. Just obtain the sorted indices. (0.25 mark)
- (e) Call the `mergeSort` method to sort the original data based on property `SAS`. Note that you should not really change `originalData`. Just obtain the sorted indices. (0.25 mark)
- (f) Call the `sumRankSort` method to take the results of the above three function calls as input and obtain the sorted indices. (0.25 mark)
- (g) Based on the result of the previous step, sort our original data, store it in variable `finalSortedData` (this variable should have the same format as `originalData`), and save the information from `finalSortedData` to a CSV file named `ZINCSubset_sorted.csv`. (0.5 mark)

Program comments: to maximize readability, you should properly comment your program. (1 mark)

### 3 Submission

- Your source code.
- A PDF printout of your source code.
- The text file `ZINCSubset_sorted.csv` that shows all expected outputs from these tasks.
- Compress the above files in a zipped folder named `COSC2P03_A2_YourFirstname_YourLastname_StudentNumber.zip` and submit it through Sakai before indicated due time.
- Late submissions will not be accepted.
- **Note: you should submit both your Java source code and your PDF printout. Missing any of them will result in a zero grade for this assignment.**

### 4 Academic Integrity

This assignment should be tackled individually. Outsourcing or teamwork is not allowed. Violation of this requirements will be seriously processed in accordance with university policies.