**Lab 1: Algorithm Exercises**

Lecturer: Laura Liu (Wen.Liu@xjtlu.edu.cn)

Teaching Assistant: Kangshi Wang (Kangshi.Wang19@xjtlu.edu.cn)

Department of Electrical and Electronic Engineering

Xi’an Jiaotong-Liverpool University

|  |  |
| --- | --- |
| Contribution to Overall Marks | 10% |
| Submission Deadline | 17th Oct, 2021 23:55 |
| To submit | * Lab1.rar/zip * A report with solution and proper description to your code. |

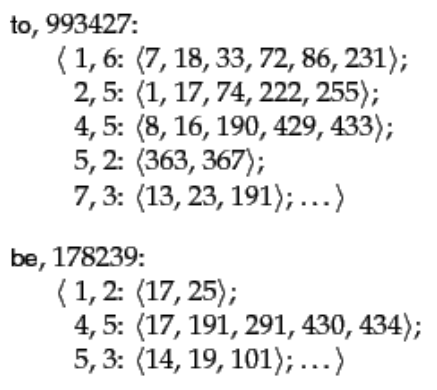
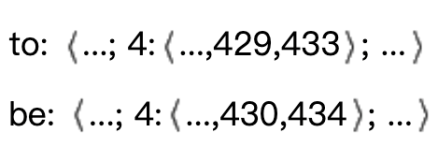
1. **INTRODUCTION**

This exercise works through some of the cases that you should understand. A large search engine has thousands of machines and can keep a ton of stuff in memory. But we should also consider small search systems, such as Spotlight search on your Mac or the equivalent Windows Search for Windows.

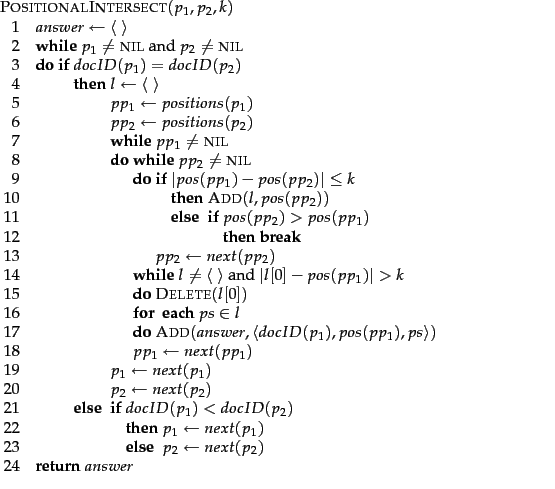
This exercise provides a skeleton for Java coding (you may download it from the module home page at Learning Mall), and all you have to do is to fulfill corresponding algorithm parts. Your report will be evaluated by the contents include your understanding of lectures, problem analyze and solve. There is no need to attach codes in the report. Note that, you should submit the Lab report as well as your Java codes in one packet (RAR or ZIP, renamed the file as StudentID\_studentname\_INT309\_Lab1.rar/zip) via Learning Mall. No hardcopy is required.

1. **TASK ONE: Bi-words Retrieval With Positional Index**

To search the bi-words–e.g., to be or to do–as vocabulary term in the search engine, “AND” query is most commonly employed. However, in order to process the queries such as “to be or not to be”–i.e., search to, be, or, not in the posting list, the positional intersection is required. The first step towards searching the phrase “to be or not to be” is to process the normal intersecting in the posting list for the term “to be”, in which the documents contain both words are found. Then we look for the first occurrence of this term and then find another occurrence of each word with a token index higher than the first occurrence. One example solution is provided in the following:

We’ve already provided in “Positional.java” a skeleton with the codes that load postings lists and try to “positional” intersect pairs of them. You should fill your own algorithm in the function of “positionalIntersect” to perform the required search. It’d be credited if you write your own algorithm that has a good trade-off on memory use/time efficiency. The following is a reference positional intersect algorithm:



1. **Notes**
2. Consider the report with the following sections: introduction, problem analysis, implementation detail, and conclusion. As a formal report, includes your name and student ID, date, etc
3. The algorithm/codes should be explained in the “implementation” part of the report.
4. Copy-paste the contents from the lab introduction file is not recommended.
5. The parameters used in the algorithm should be well defined.
6. Font: Times New Roman; font size: 12; line spacing: 1.2; Maximum: 4 pages (A4).
7. Marking Scheme：Report 50%+Codes 50%