School of Information Technology, Deakin University

SIT320 — Advanced Algorithms

Pass Task: **Advanced Hashing and Sorting**

At the completion of the module (**Module: Advanced Hashing and Sorting**), you are required to fill a lesson review by doing following activities.

Your tutor will then review your submission and will give you feedback. If your submission is incomplete they will ask you to include missing parts. They can also ask follow-up questions, either to clarify something, or to double check your understanding of certain concepts.

Task List

- **(Note: There is no module write summary for this task.)**

For some of the following activities, we have provided you some Python code. You can download it from the "resource section" of the module. This code is only provided for those students who are new to Python and for tutors to better support them. You are welcome to have your own implementations of the algorithms built from scratch.

- **(1)** Given a bucket of size 3, hash following values with extendible hashing: 16, 22, 26, 20, 3, 1, 12, 11, 13, 19, 38, 47, 46. Use ASCII representation as the hash code.
- **(2)** We have provided you with the code for extendible hashing. Run the code with various use-cases so you can demonstrate the ownership of the code. Provide code reflections in following forms:
  - a) Evidence of running the code with various use-cases — e.g., changing the number of buckets, reading data from external files, simulating data generation, etc.
  - b) Modification of the code to obtain different behaviour — e.g., use of dictionaries instead of lists, refactoring code into various functions or classes, etc.
  - c) Detail comments of various lines of the code explaining which part of the algorithm it is executing, this can be in form of detailed inline comments or adding paragraph cells in the juypter notebook.
  - This is a third-party code, we welcome you to do a complete overhaul of the code or write the code from scratch.
  - Your tutor will ask you various questions about the code and its inner working, you should be able to explain it.
- **(3)** We studied various sorting algorithms in SIT221. In python, you can access various sorting by just doing np.sort (https://numpy.org/doc/stable/reference/generated/numpy.sort.html). In this activity, you will be writing code for external MergeSort. You will simulate the idea of

pages in form of lists. Each page has a limit on number of records they can store. This will be translated into limit on the size of the list. E.g., you can specify the maximum list size to be of N. Specify a buffer of size n — which means that you are expected to only load only n - 1 lists in the memory for reading and 1 for writing. You are expected to store lists on the disk and write in the same file. Your data on the disk can be of the format:

  - 0: [19, 20, 87, 95]
  - 1: [55, 1, 22, 20]
  - 2: [87, 34, 12, 17]
  - 3: [30, 88, 98, 2]
  - ……
  - 99: [44, 33, 90, 1], etc.

- Note, you are expected to generate your own data using random number generation in numpy.
- It is okay, if you want to load entire data into the main memory, rather than repeatedly writing on the file, but your algorithm should mimic external merge sort, that is you have a conquer pass, and then merging of various lists systematically.