CSCI 230 -- PA 2

More on Hashing

Feel free to discuss and help each other out but does not imply that you can give away your code or your answers! Make sure to read all instructions before attempting this lab.

You can work with a lab partner and each one must submit the same PDF file (include both names in the submission file). Each person must include a brief statement about your contribution to this assignment.

You must use an appropriate provided template from Canvas and output "Author: Your Name(s)" for all your programs. If you are modifying an existing program, use "Modified by: Your Name(s)".

Question 1: What are some effective techniques to generate hash code for a key of data type string?

Question 2: Why do we need to use compression functions? What are some common compression functions?

Exercise 1 – Perform P-9.9 (pages 420-421 of C++ book). Use 37, 40, and 41 for parameter **a** with polynomial hash codes or a shift of 1, 5, and 13 for cyclic shift hash codes (only need to pick one method). Test it on the following file, US Declaration of Independence, and you can assume that words are separated by white space characters. Output number of words and number of collisions for each case. Make sure the results are reasonable (one of the 3 values should give a much higher number of collisions).

Exercise 2 – Create a program to collect some data about chain hashing that you worked on in previous PA. You would be able to enter the name of input data file and a load factor. Each input data file will have N records and the first value in the file will tell you how many records are in the file. You would need to use N and the load factor to determine the size of the hash table. **Do not** rehash the table like the version in the book (i.e., need to modify the code from the book). The first value of each record will be the key (county/state code as integer type) and remaining items on each record will be the value (population and county/state). After all the entries are inserted to the table, print the **table size, average number of probes, and maximum number of probes for the worst case**. It would take at least one probe for each insertion (checking initial location). Therefore, it would be two probes if second location is examined.

After confirming that your implementation works reasonably well against the small input file (popSmall.txt), run it against the larger input file (popLarge.txt) and collect some required data. You must try the load factors of 0.25, 0.5, 0.75, and 0.9 against the large input file (4 different runs). Make sure that your hash table size is determined correctly according to the load factor. For example, if a file has 5 values and for a test at load factor 0.5, the table size is 11 (must determine and use nearest prime integer larger than 10 for this case).

You can only earn extra credit for one option below.

Extra Credit option A: Collect data for second hash code for exercise 1.

or

Extra Credit option B: Collect same data as exercise 2 for open addressing with double hashing and compare results to exercise 2. You can use ProbeHashMap class in Java book.

Fill out and turn in the PA submission file for this assignment (save as PDF format).