CSCI 230 -- Project 1 (Hashing)

Due Wednesday, 03/18/2020 (MW Section)
Due Thursday, 03/19/2020 (TTh Section)

This project is designed to help you understand hashing, its implementation, its performance, and its usage. Define and implement three hashing schemes as three separate classes, **Chain Hashing, Linear Probe Hashing**, and **Double Hashing**, as discussed in the textbook and class. You should take advantage of inheritance when possible and you may want to try one hashing scheme first (Chain Hashing should be mostly implemented via the lab already).

Create a program to test and perform simulations on the three hashing schemes by using those three classes. You would be able to enter the name of input data file, hashing scheme, and 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. 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. It then provides the following menu (like project 4 in CSCI 220):

1. Search for a record
2. Insert a record
3. Delete a record
4. List all records
5. Exit

Your program will continue until the user wants to exit this program. For option 1, user will be prompted for the county/state code; if the record is found, display complete record. For option 2, user will be prompted to enter the population record on one line: county/state code, population, and county/state name (if a key exists then you need to update the value part). For option 3, user will be prompted for the county/state code. For option 4, you need to list all records and display one record per line (order by index/address of the table). In addition, your program must display number of probes for options 1 to 3.

Run the following test cases after the hash table is constructed with ***p1small.txt***, a load factor of 0.25, and each of the three hashing schemes (3 separate runs):

1. List all records

2. Search for 6037
3. Search for 6000
4. Insert 6066, 1, “New County, CA”
5. Insert 6065, 2000, “Riverside, CA”
6. Delete 6999
7. Delete 6075
8. Delete 6055
9. List all records

After confirming that your implementation works reasonably well against the small input file (p1small.txt), run it against the larger input file (p1large.txt) and collect some required data. You must try the load factors of 0.25, 0.5, 0.75, and 0.9 for each hashing scheme against the large input file (12 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). Make sure to **annotate each result** so that we know which input file, hashing scheme, and load factor.

Provide a report on your findings on the large input file and make sure to relate them to the results in the textbook if applicable. You must summarize your results using one or more table. In addition, include any unexpected results and discuss them.

Team Project Option: It is not required but you can work in a team of two members. You must sign up by Wednesday, 03/04/20, for MW class or Thursday, 03/05/20, for TTh class if it is going to be a team project. Make sure to split the responsibilities equally and work together. You do need to provide a short description on who did what and your experience as a team project. It is possible that all team members may not receive the same score. **If it is a team project, you need to implement at least one of extra credit options as a required component and you can optionally implement the other one for extra credit**. For an individual project, you can pick one of the two extra credit options and you can only receive at most 5 extra points.

Extra Credit 1: You can earn up to 5 extra points if you can report the average number of probes and maximum number of probes for both successful and unsuccessful searches (collect separate data for the two situations). Be sure to try enough cases so it is easiest to generate and utilize another input data file (you need to generate a new data file and this data file should have about 50 keys in p1large.txt and about 50 keys not in p1large.txt). Compare the results between the two situations (successful and unsuccessful searches) and to the data collected from the initial insertions.

Extra Credit 2: You can earn up to 5 extra points if you can print the **number of clusters, average cluster size, and size of the largest cluster** after insertions of all entries into the table. A **cluster** is defined to be two or more consecutive elements for open addressing (including wrap around) and two or more elements in a chain for chaining. You must collect data for load factors of 0.25, 0.5, 0.75, and 0.9 for each hashing scheme against the large input file (12 different runs). You also need to discuss the correlation between clusters and probes for different hashing schemes.

Please provide documentation and applying good coding style because it is part of the grade. You must come up with enough test cases since the test cases are also part of the grade. Please submit the following items (items 1 – 5 can be either be hardcopy or one pdf file on Canvas; if all items are submitted via Canvas, you must turn in a hardcopy of either title page or project sheet):

1. Title page with name, class, project number, and relevant information about your program (compiler and system used, file names).
2. Notes about your program (status of your program and lessons learned).
3. Summarized results in a table and discuss collect data **(5 points as part of documentation)**.
4. Printouts of any input/output.
5. A printout of the source code.
6. A copy of your source code on Canvas (.h /.cpp or .java file).

Your program will be graded as follow:

- Correctness/Efficiency: 35 points
- Test Cases: 5 points, Documentation/Coding Style: 10 points

---