How Good Is Your Hash?

Performance Testing Lab

Description.

In this project, you will measure the performance of your hash table by adjusting system parameters and measuring the time needed to perform insertions and searches. Your results should give you an understanding of tradeoffs involved in hash table design. Read the notes on hashing before you start coding!

Work with a partner for this project – "Divide and conquer." While you may work on separate parts of the program initially, when finished combine all code into a single project for your final test runs. Turn in your Java program, results text file, & Excel spreadsheet together.

Measure the performance of your hash table under these conditions – this lab will have a total of 42 test runs.

- Use 4 different collision resolution schemes
  - Closed hashing
    - Linear probing
    - Quadratic probing
    - Random probing
  - Open hashing
    - Chaining with linked lists

- Use 2 different hashing functions for each collision resolution scheme
  - One you think will work well
  - One you think is poorly designed (why do you think that?)

- For each combination of collision resolution scheme and hash function, use these values of load factor, α
  - For test runs that use closed hashing for collision resolution
    - Use α: 0.10, 0.50, 0.80, 0.90, 0.99
  - For test runs that use open hashing for collision resolution
    - Use α: 0.10, 0.50, 0.80, 0.90, 0.99, 2.00

Note: Records in the provided data files have unique names – if you get a name match, you don't have to check the phone number to be sure it's the correct record (of course if you're selling this program, you have to check!)

Program Organization

Do this once at program start up:

- Create an *ArrayList* of *String*s, with an initial size of 50K.
- Open the "Large Data Set.txt " – called *store* below (50K records)
- Read a line in the text file & add the string to the *ArrayList*
    - Repeat to the end of the file
- Close the file.
- Build similar *ArrayList* objects for the "Successful Search Records.txt" file and the "Unsuccessful Search Records.txt" file

For each test run, do the following:

1. Given an input file of 50k records, build a table sized for the load factor to be profiled
    a. e.g, for an $\alpha = 0.50$, the table size should be ~100K
    b. e.g, for an $\alpha = 0.67$, the table size should be ~75K

2. Start "Build Table" timer
3. For each record in the "Large Data Set.txt " (large file)
    a. Parse the record
    b. Build an object
    c. Insert it into the hash table
        i. For open hashing (chaining with a linked list)
            1. If first insertion, create list & add to list
            2. If collision occurs, increment collision counter & add to list at that location
        ii. For closed hashing (linear, quadratic, random)
            1. Increment collision counter for each unsuccessful probe
4. Stop "Build Table" timer

5. Start "Successful Search" timer
6. For each record in "Successful Search" data set
    a. Parse the record
    b. Build a Name object
    c. Search the table – get number of probes needed to find the entry
        i. Count number of objects checked before correct object is found
7. Stop "Successful Search" timer

8. Start "Unsuccessful Search" timer
9. For each record in "Unsuccessful Search" data set
    a. Parse the record
    b. Build a Name object
    c. Search the table – get number of probes needed to determine the record isn't in the table
        i. Count number of objects checked before reaching a null entry (closed hashing) or the end of the hashed list (open hashing)
10. Stop "Unsuccessful Search" timer

11. Print a report to a text file containing:
    a. Type of hashing used (e.g., linear probing, chaining)
    b. Hash function used (your descriptive designation)
    c. Number of records added to the table, table size, and load factor

d. Average insertion time
e. Number of table insertion collisions
f. Number of collisions vs. number of insertions (expressed as %)
g. For open hashing, print average list length
    i. (number of insertions) / (number of insertions – number of collisions)
h. Average time & number of probes needed to find table entry
i. Average time & number of probes needed to determine entry is not in table

Note: all test runs results should be output to a single text file.

Reults Spreadsheet.

Transfer data from the program output text file to Excel – yes, use Excel!

Make the following charts:

1. 3D bar chart: average insertion time for each collision resolution scheme vs. load factor α
    a. Use test run data from the best hashing function
    b. Order the axes so all result bars are visible (shortest bars in front)

2. 3D bar chart: collision % (on insertion) for each collision resolution scheme vs. load factor α
    a. Use test run data from the best hashing function
    b. Order the axes so all result bars are visible (shortest bars in front)

3. 2D line chart: for open hashing, average list length vs. load factor α
    a. Use test run data from the best hashing function

4. 2D line chart: for *successful* search, number of probes vs. load factor α
    a. Use test run data from the best hashing function
    b. Similar to textbook chart in notes – you will have 4 collision resolution methods
    c. How do your results compare?

5. 2D line chart: for *successful* search, average time vs. load factor α
    a. Use test run data from the best hashing function
    b. Similar to textbook chart in notes – you will have 4 collision resolution methods
    c. Do the average time results tell you the same story as the number of probes results?

6. 2D line chart: for *UNsuccessful* search, number of probes vs. load factor α
    a. Use test run data from the best hashing function
    b. Similar to textbook chart in notes – you will have 4 collision resolution methods
    c. How do your results compare?

7. 2D line chart: for *UNsuccessful* search, average time vs. load factor α
    a. Use test run data from the best hashing function
    b. Similar to textbook chart in notes – you will have 4 collision resolution methods
    c. Do the average time results tell you the same story as the number of probes results?