Employee Database – Part 2

Performance tests

You will implement performance tests below in a new **Tester.java** (performance test). Simply coding a hash table is "good enough" at this point, but *understanding* why and how hash tables work, and the trade-offs in memory / performance is important for an aspiring computer *scientist*.

You will use the data in the **"Large Data Set.txt"** file that contains names and IDs. You will also use the files **"Successful Search Records.txt"** and **"Unsuccessful Search Records.txt"** that contain entries that do / don't exist in the original data set to test the performance of your hash table (make sure you look at the files so that you understand the format of the input). You will perform the following tests:

1. Read the contents of the text files into your program, store in a convenient data structure.

2. For each combination of collision resolution scheme / hash function (e.g. linear probing / good hash function), use these values for α (where load factor α is number of entries per number of buckets – not sure what is a load factor? – Google my friend).

   - α: 0.10, 0.50, 0.80, 0.90, 0.99

   Given the input file of 50k records, build a table sized for the load factor to be profiled.

   a. e.g. for an α = 0.50, the table size should be ~100k
   b. e.g. for an α = 0.67, the table size should be ~75k

3. Start "Build Table" timer `//the System class contains a method that will return the current time`

4. For each record in the "Large Data Set.txt":

   a. Parse the record, build an object, insert it into the hash table

      i. Increment collision counter for each unsuccessful probe

5. Stop "Build Table" timer

6. Start "Successful Search" timer

7. For each record in "Successful Search" data set:

   a. Parse the record

   b. Search the table – get number of probes needed to find the entry

        i. Count number of objects checked before the correct object is found

8. Stop "Successful Search" timer

9. Start "Unsuccessful Search" timer

10. For each record in "Unsuccessful Search" data set:

    a. Parse the record

    b. 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 the end of the hashed list

11. Stop "Unsuccessful Search" timer

12. Print a report to a text file containing:

    a. Type of hashing used (e.g., linear probing)
    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. Average time & number of probes needed to find table entry
    h. 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.

NOTE: Records in the provided data files have unique names – if you get a name match, you don't need to check the ID to be sure it's the correct record (if you were selling this program, you would have to check!).