CS151:Spring 2023

Lab 9

1. Download `Entry.java`, `Map.java`, `AbstractMap.java`, `AbstractHashMap.java` and `ProbeHashMap.java` from `~dxu/handouts/labs/09`. This is the book's implementation of a linear-probing hash table.

2. Study `ProbeHashMap.java` and make sure you understand the implementation details.

3. Download `dictionary.txt` and `search.txt`. You will write a program that first creates a `ProbeHashMap`, and then insert words in `dictionary.txt` one by one as keys into that hash table. Values are the same words. There are 24520 words in the file. Your initial hash table capacity should be chosen as a reasonably large prime number.

4. As you insert each word, compute the average number of probes and maximum number of probes. Number of probes is the number of times an open-addressing hashtable needs to try before finding the correct array location to either insert or find a key-value pair. With no collision, the average and max number of probes per key should be 1. Print these values after all insertions are completed. Note that, to be able to compute these values, you will need to update `ProbeHashMap` class. The lines of the output should print the hash table statistics in the following order.

   ```
   average number of probes during insertions:
   max number of probes during insertions:
   load factor after insertions:
   ```

   The load factor is the number of elements in the hash table divided by the capacity of the hash table. It indicates how full the hash table is.

5. Now, read each word from `search.txt` (one word per line) and search for each one in the hashtable. If the word is found, just print it back out. If the word you are looking for is not in the dictionary, assume that it is misspelled. To suggest corrections, for each misspelled word, list any words in the dictionary that are obtainable by any of the following rules:

   - Change one letter: For example, if the misspelled word is "kest", try all possibilities of changing one character at a time, and look the modified word up in the dictionary. The possibilities will be "aest", "best",...,"zest", "kast",...,"kzst", etc. Assume that the words in the dictionary file are given in lower case. So, before looking up a word in the dictionary, convert the word to lower case.
   - Exchange adjacent letters: For example, if the misspelled word is "ebst", try "best", esbt" and "ebts".
   - Remove one letter: For example, if the misspelled word is "tbird", try all possibilities of removing one letter at a time, and look the modified word up in the dictionary, which are: "bird", "tird", "tbrd", and "tbir".

Note that, you have to try all possibilities, but only the ones that are actually found in the dictionary are possible corrections to be suggested.

6. **Output Format:** For each misspelled word, generate a single output line as follows. First print the word itself, then colon(:), and finally the list of possible corrections separated by commas. For example, if the misspelled word is "kest", the output may be:
   `kest:  best, fest, jest, lest, nest, pest, rest, test, vest, west, zest, est`

7. As you search each word/possibly corrected word, compute the average number of probes, and maximum number of probes. Print these values once after all searches are completed.