Cairo University
Faculty of Engineering
Computer Engineering
Fall 2020

Design and Analysis of Algorithms
Lab 3
Hashing

Objectives

After this lab, the student should be able to

- Implement hashing

Requirements

1. **Linear Probing**

Given the file words20k.txt (one word per line, all words unique, 20,000 words in the file):

1) Read the words one by one into a string array A.
2) Create another array (H) of size 24,000, fill with empty strings (i.e. "") first, and then copy items from A into H, using a hash function and linear probing for collision resolution.

The hash function to use, for string s:

- h'(s) = 39 * int(s[0]), if s.size() == 1
- h'(s) = 39 * int(s[0]) + 392 * int(s[1]), if s.size() == 2
- h'(s) = 39 * int(s[0]) + 392 * int(s[1]) + 393 * int(s[2]), if s.size() == 3
- h'(s) = 39 * int(s[0]) + 392 * int(s[1]) + 393 * int(s[2]) + 394 * int(s[3]), otherwise, and then:
- h(s) = h'(s) % 24000;

3) Measure the average item insertion time for the first 500 inserted words, for the next 500 inserted words, etc. until the last 500 inserted words.

4) Read 1000 words from array A, from index 14000 to 14999. For each word, search and delete it from the hash array H. Track the number of probes needed for each of these 1000 words and use it to find the min, max and average number of probes across the 1000 searches.

2. **Secret Sentence**

Unfortunately, you are now trapped and your only exit is to know the secret sentence that will allow the TA to let you out. The secret sentence can be found inside the book "Tale of Two Cities" that can be found in the file *"two-cities.txt"*.

The secret sentence can be formed from the most frequent words in the book in the following order {11, 23, 22, 43, 3, 47}. To clarify, the first word in the sentence is the 11th most frequent word in the book, the second word is the 23rd, and so on. Implement the program with the lowest time complexity to get you out of the door.

Helper functions are provided that reads the file, removes punctuation and converts to lower case. All you have to do is **implement** the function ***findSecret*** that takes a vector of tokens (words from file) and returns a string representing the secret sentence. Hint: you can use **std::unordered_map** and **std::sort** for this question only (unorded_map is implemented internally using hash tables).

Submission Notes

Submit **1 zip file** containing **two cpp files** (one for each problem) in the following format:

firstName_secondName_ID.zip
    1_firstName_secondName_ID.cpp
    2_firstName_secondName_ID.cpp

Example:

John_Doe_1100000.zip

  1_ John_Doe_1100000.cpp

  2_ John_Doe_1100000.cpp

Do not submit any other file but these two files.