

Winter Quarter, 2015 Programming Project #1

1. Introduction

The purpose of this project is for you to practice the principle of “hashing” – a method of assigning elements of a collection to a particular sequential structure (e.g., array). A popular interpretation of hashing is, given a collection of values $V = \{v_1, v_2, \dots, v_n\}$, create the pairing $(key, value)$, where $key \in K$ ($K = \{k_1, k_2, \dots, k_m\}$) such that:

1. Regardless of how “messy” (e.g., variable length string) each v_i is, one can quickly (ideally, in constant time) find its corresponding k_j .
 2. The function $h(v_i) = k_j$ should have some “nice properties”:
 - a. If the purpose is to solely generate key-value mappings, h should be “easily computable” whereas, in encryption scenarios, it should be hard to “crack”.
 - b. It should map the elements of V **uniformly** among the set of (typically integers) values of K .
 3. If the settings involve some “dynamics” (e.g., the size of V grows over time), h should be easily adjustable to cater to a new (larger in size) K .
- etc...

In this project, you will need to implement a “comparative analysis” of two hashing methodologies (see below) in two different settings. Namely, you will test the behavior of the hash functions when you have:

1. Array of size 100.
2. Array of size 200.

to distribute the values¹ (entries).

One of the hash functions will be:

$$h_1(X) = \sum_{\text{each character } c \in X} \text{ASCII}(c) \% \text{size}$$

where:

X is a string and “size” is the available size of the array used (100 or 200);

The other hash function should be $h_2(X) = X.hashCode()$ (i.e., the readily available function from JVM).

Your application should perform as follows:

1. Ask the user for the size of the array (100 or 200);
2. Open the file “input.txt” (a practice file will be provided for you);
3. Generate two files: “output1.txt” and “output2.txt” where:
 - a. Lines will start with a number corresponding to the integer values 1-100 (or, 1-200);
 - b. Consecutive lines will be separated by an extra blank/empty line;

¹ OF course, you are welcome to use ArrayList.

- c. If there is a collision (e.g., both “abc” and “1ab2c...” strings end up with the same hashing value), the later of the two will be added in the same line as the previous one, separated by a comma and space.
 - d. If no string from the input has been mapped to a particular line, that line should (by default) contain “EMPTY LINE...”
4. Come up with a metric (measure) to determine which hash function performs better. Implement your metric on the sample data and output it the result as a file.
5. Think of a new hash function different than the above two. Implement your suggested method and output your results and metrics as in item 4. (Extra credit if your hash function is a better than the two mentioned in this project).

2. Hints

1. Finish everything in-memory first, then simply write the output files;
2. Ideally, one should use a linked list where each node contains a string, for which the “head” will be the corresponding element in the array (NOTE: we assume chaining as a solution for collisions, not open addressing). However, life will be made much easier if each element of the array is a string variable, where the “collided” elements will be appended after a comma;
3. A note on hashCode(): it will return to you a numeric (signed) value. Hence, you need to “scale” it down to 100 (or 200), by taking the respective “%”

3. Other Remarks

This project can also be done in teams of up to 3 people.

When submitting this programming assignment:

- Put *all and only the source code* files in a folder that will be named as your *classID*;
 - o Use the classID of only one person of the team (i.e., one submission per team);
 - o Provide a plain-ascii (.txt) file describing some exceptions that you thought of, and list the names of the team-members (in the same folder as the source files);
- Zip the Folder and upload it to Canvas.