CSCI 115 Lab

Week 9- Hash tables

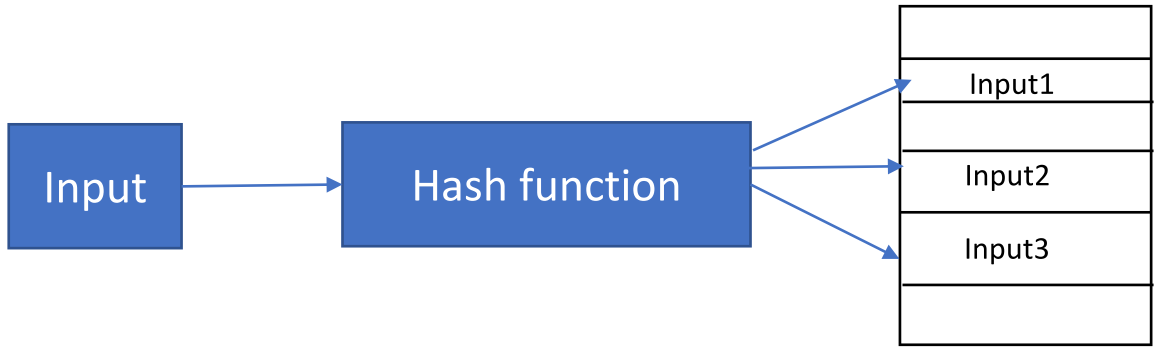
- Professor: Dr. Matin Pirouz
  Email: mpirouz@csufresno.edu

- TA: Shreeja Miyyar
  Email:shreejarao12@mail.fresnostate.edu

Table of Contents

Hash tables

- A hash table is a data structure which is used to store key-value pairs.
- A hash function is used to compute the index into an array in which an element will be inserted.



- The hash function calculates an index for each input and that input is stored in that index of the hash table array.

For e.g, consider the inputs - Input1, Input2, Input3. If we have a hash function H(x), such that H(x) produces a number or index.

- H(input1) = 1 -> Input1 is stored at index 1 of the hash table array.
- H(input2) = 3 -> Input2 is stored at index 3 of the hash table array.
- H(input3) = 4 -> Input3 is stored at index 4 of the hash table array.

Hash functions

- Hash function maps an input (numbers, strings) into a small integer value that can be used as an index in the hash table array.
- A good hash function should be efficiently computable and produce a different index for different inputs. A bad hash function produces the same index for different inputs which will result in hash collision.

<u>E.g. for hash function</u>

- Consider a hash table array with size 5 and input strings ”ABC” and “XYZ”. We need to map the input strings to an index in the hash table array.
- The hash function designed for this is:

H(x) = (sum of ascii value of each character in the input string) % Hash table size

H(“ABC”) = (Ascii(“A”) + Ascii(“B”) + Ascii(“C”)) % 5 = 3

H(“XYZ”) = (Ascii(“X”) + Ascii(“Y”) + Ascii(“Z”)) % 5 = 2

Hash table = [Nil, Nil, “XYZ”, “ABC”, Nil]

Reference for ASCII value of characters: http://www.asciitable.com/

Hash collisions

- Hash collisions occur when the the hash function returns the same number/index for two or more inputs.
- When there is a collision, the new input cannot be inserted to the index generated by the hash function and we need to come up with a method to handle these kind of collisions.
- Hash collisions occur due to bad hash functions which fail to produce unique index for different inputs.

<u>E.g. for hash collisions</u>

- Consider a hash table array with size 5 and input strings ”ABC” and “XYZ”. We need to map the input strings to an index in the hash table array.
- The hash function designed for this is:

H(x) = Length of the input string % Hash table size

H(“ABC”) = Length of string ABC % 5 = 3

H(“XYZ”) = Length of string XYZ % 5 = 3

Here you can see the hash function has produced the same index for string ABC and XYZ which resulted in collisions.

Handling hash collisions

- **Chaining**: In this method, each cell of the hash table points to a linked list of inputs having the same hash value. This requires additional memory to store the inputs with the same hash values.
- **Open Addressing**: In this method all the inputs are stored in the hash table itself, and each slot in the hash table is probed whether it is empty or full. There are two methods of probing.
    - *Linear probing*: In this method we linearly check for the next empty slot in the hash table.
    - *Quadratic Probing*: In this method we check for next i^2 empty slot in the hash table.

E.g. for Linear probing

Consider a hash table = [input1, input2, Nil, Nil, Nil]

Now we need to insert "input3" to the hash table. The hash function produced index 0 for this input, i.e.

H(input3) = 0.

Since index 0 is not empty we check the next slot, i.e. H(input3) + 1 = 1. We find that index 1 is also not empty.

We then check the next slot H(input3) + 2 = 2. The index 2 is empty and input3 is stored at that position in the hash table.

Hash table = [input1, input2, input3, Nil, Nil]

Lab Assignment

<u>Hints:</u>

- Since it is mentioned that number of unique words are 3684, you can define a hash table array of size 4001.
- For this assignment you can use the following hash function;

  H(x) = *(sum of ascii value of each character) % HASH SIZE*

  HASH SIZE = 4001

- You can use an integer variable to keep track of number of array accesses for each word to be inserted into the hash table array.
- Add ”iostream” and “fstream” headers in the cpp file to to be able to read files.
  - #include <iostream>
  - #include <fstream>

Coding guidelines

- In the main function read the words from the input txt file.
- Create a function update() which takes a string argument and calculates the hash value that string and adds it in the hash table.
- Create a function size() which will check if the hash table array is empty or not and returns the count of non-empty slots in the hash table which will return the number of unique words in the input file. You can also use a global integer variable to keep track of number of unique words in the txt file.

Questions?