Wesleyan University, Fall 2021, COMP 211
Lab 11: Hash tables

---

1. Overview

The goal of this lab is to is to gain some practice working with hash tables, using them to implement a set abstract data type and see how you would extend what you've done to a dictionary abstract data type.

2. Create and organize your program files

For this lab, you should create a new directory lab11 in your labs directory to hold your code. Then do the following.

(1) Download the following files from the lab website. During this lab you will modify both `set.c` and `set.h`.

```
Makefile
driver.c
set.c
set.h
hash.c
hash.h
```

3. Set Abstract Data Type

You will implement a set abstract data type using a hash table that uses separate chaining. Recall that a hash table using separate chaining is implemented using an array of linked lists. An array of size `M` will keep track of `M` linked lists: the ith element in the array will correspond to the dummy head node for the ith linked list. In `set.h`, `M` is named `NUM_BUCKET` and has already been initialized for you to equal 101.

There are a number of operations that are natural to perform on a set such as adding elements and deleting elements. The signatures and specifications of the functions you must implement are given in `set.h`, which also specifies that there is a structure type `set`. This structure is defined for you in `set.c`; and that is also where you will implement all the functions specified in the header file (we have written stubs for those functions for you that you should replace). The hash function that you must use is already implemented for you in `hash.c` as is a function that checks the invariant we described in class. Every function that creates or modifies a `set` argument must verify that the modified `set` is valid before returning.

Although `NUM_BUCKET` has been initialized in `set.h` to 101, you may wish to set it to a much smaller value when testing your code in order to make it easy to create collisions.

4. Dictionary Abstract Data Type

You may have come across previously a dictionary (also known as a map) abstract data type. This can also be implemented using a hash table that uses separate chaining. The dictionary or map abstract data type is similar to a set, but now associates a value with each key. Thus, a

dictionary is a set of key-value pairs, rather than just keys. Like a set, a dictionary is an unordered collection in which no key occurs more than once, although the same value may be associated with different keys. Another perspective on a dictionary is that it is a function from keys to values.

There are a number of operations that are natural to perform on a dictionary, such as adding a key-value pair and looking up the value corresponding to a key. Think about how your set code would change in order to instead implement a dictionary.

5. Going further

The hash function that you will use is based on a random number generator. This hash function relies on the fact that for the same random seed, the random number generator produces the same sequence of random numbers. The hash function then works as follows, for a given key. First, the random number generator is seeded with a key. Then the random number generator is called to generate one random number `r`. The index (or bucket) into the hash table is then computed using `r mod M`. Thus `hash(key) = r mod M`.

The hash table implementation that you write in this assignment are missing an important feature of "industrial-strength" implementations. In order to ensure constant average-case insertion, deletion, and membership testing, we have to fix some number K, and then ensure that the average chain length is always $\leq K$. This is impossible if there are more than $K \cdot$ `NUM_BUCKET` elements in the set. The solution is the following: when the size of the set reaches this size, allocate a new, larger array of chains, and redistribute the keys into the new array.