Assignment 6 Writeup

Pehara Vidanagamachchi

November 2022

1 Main Program

The purpose of this project is to create a bloom filter and hash table that helps create a functioning ban hammer program. The program reads a list of old speak and newspeak, alongside bad speak, which is just a translated version of old speak using newspeak. Each word checked must be added to the bloom filter and checked with the hash table. Essentially the filter is looking to conserve social cohesion and avoid public distress in its make-believe setting. There are message formats to follow in the scenario a citizen uses inappropriate language deemed as unfit due to its hurtful, offensive, unfortunate and far too descriptive language. In order to create this within my code I had to create multiple helper programs such as bloom filter which filters to see whether a word is a member of a set or not. This was accompanied by a bit vector which helped allocate memory. We were given a city file that contained hash functions to use within our code which is used for hashing shorter strings. Alongside this, we had linked lists which acted as a solution towards avoiding hash collisions with the use of nodes. Nodes are another function I created that helps differentiate between all the different words and simplifies the implementation of the link list. Finally leaving me with the parser which acted as a way to parse out the words that the citizens speak and transform them into an input stream. All of these functions combine to create the main program which then executes the task given.

2 Pre Lab questions

- **(a) How does the number of Bloom filter bits examined per miss vary (for the same input) as the Bloom filter varies in size? -** By doing this assignment I discovered that the lower the bloom filter was, the higher number of bits examined per miss also increased. By comparing this with the statistics and the bloom filters usage of a simple bit vector, it was apparent that there were frequently more misses when there are fewer index values. This ultimately means that the number of bloom filter bits

examined and the bits examined per miss are inversely proportional when the size is increased.

- **(b) How does changing the Bloom filter size affect the number of lookups performed in the hash table? This is related to the false positive rate of the Bloom filter; we discussed this in the lecture that discussed Bloom filters. -** With the use of my code, I noticed a pattern where, as the bloom filter would decrease in size, the probability of bad speak and old speak having the same index increased. This ties back to false positives since this produces a false positive when the word is unable to be located within the node and linked list. So when the Bloom filter is faced with a smaller size than its counterpart, there is a higher risk of a false positive appearing.

- **(c) How does the number of links followed without the move-to-front rule compare to the number followed with the move-to-front rule? -** My code displays that the number of links followed without the move to the front rule compared to its counterpart where the number followed with the move to the front rule is increasing. With the additional parameters of the move-to-front rule, my code witnessed a jump in links due to the different movement patterns.

- **(d) How does the number of links examined vary as the size of the hash table varies? What does this say about setting the size of the hash table when using a chained hash table? -** The number of links examined varies with the size of the hash table as it changes as more information is given. Setting the size of the hash table helps adjust the workload which then changes where the keys are allocated which also decreases the number of collisions that occur.