Pre-lab #1

```
1. Insertion (*bf , key) {

       hash ( salt [0], key ) % size

       bv_set ( hash () )   // set bv

       probe ( hash () )   // check if it's set
         x
         3      // do it three time
                   with different hash
                   functions to reduce false
                   positive.

                // we set totally of three
                   bits.
```

2. Creating bloom filter:

Time : $O(m)$ // take linear time

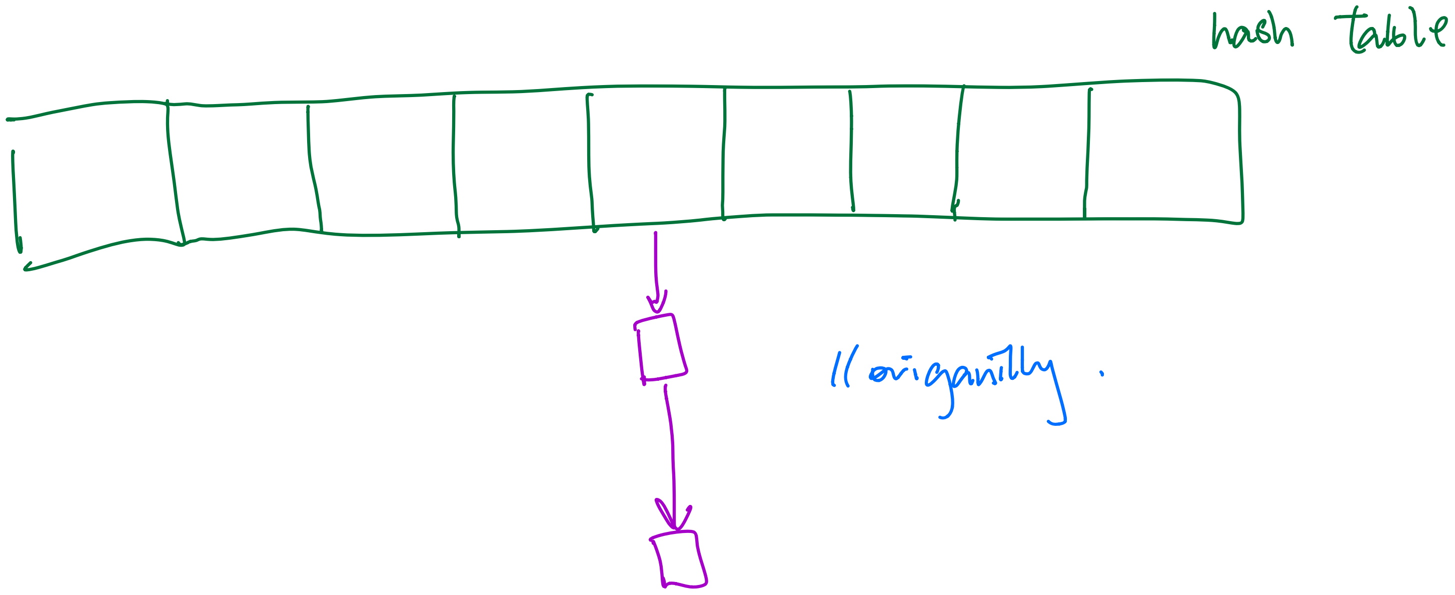
Space : $O(m)$ // take linear space

Set bit

Time : $O(k)$ // we need to hash k time

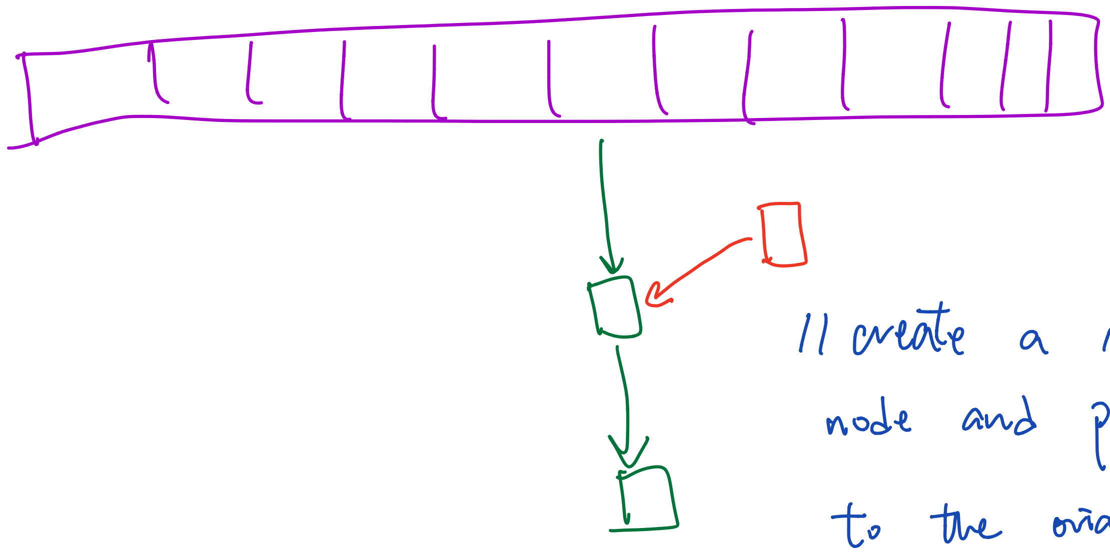
Space : $O(1)$ // no matter how many hash function we have, we only need to calculate the hash bit and set it into the bloom filter.

Pre-lab #2

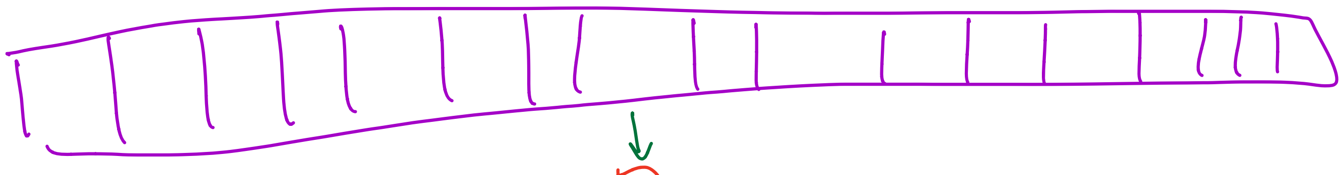


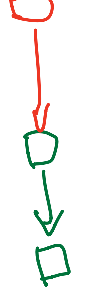
// origanilly.

// to insert a new element to the list on the top.



// create a new node and point to to the original node





// replace it in the hash table

2. pseudocode

```
insert ( ) {
    index = hash (word) // find what index this key belongs to.

    if ( lookup(index)) { return } // we dont need to do anything if the word is in the link list already
else :
    listNode new ; // new object.
    new. next = hashtable [element] .next // point the new node to the old node .
hashtable [element].next = new
    // replace the old node .
```

Flowchart. - How To Use

- `./hatterspeak -s` will suppress the letter from the censor, and instead print the statistics that were computed as illustrated below
  * `Seeks:` number of seeks performed
  * `Average seek length:` links searched / total seeks
  * `Average Linked List Length:` average length of non-zero linked lists in hash table
  * `Hash table load:` percentage of loading for the hash table
  * `Bloom filter load:` percentage of loading for the bloom table
- `./hatterspeak -h size` specifies that the hash table will have `size` entries (the default will be 10000).
- `./hatterspeak -f size` specifies that the Bloom filter will have `size` entries (the default will be 2^{20}).
- `./hatterspeak -m` will use the *move-to-front rule.*
- `./hatterspeak -b` will not use the *move-to-front rule.*
- ANY combination of these flags except for `-m -b` *must be supported.*

blood filter.

add all word in oldspeak.Txt.

add all oldspeak in hatterspeak.Txt.

↓

if word shows positive in bloom filter,

go to hash Table.

↓

Hash Table

1. check if the word exist.

2. if so, see if the word has a translation.

Hash Table

1. Find the existing oldspeak in the Hash table.

2 Check if the word has an associate hatterspeak with it.

$$\begin{tikzcd}
\underline{\text{if found}} \arrow[d] & \underline{\text{if Not found}} \arrow[d] \\
\text{print translation} & \text{Send to dungeon:}
\end{tikzcd}$$

"%s _ %s /n" old, new

How to program

1. Create a bloom filter, and a Hash Table.

bloom filter

1. put in all oldspeak.txt.

```
while ( fscanf (fp, "%s\n", eachword) != EOF)
```

2. put the first world of hatterspeak.txt.

```
// the first word is oldspeak.
// the second word seperated by a space
   is hatterspeak translation.
```

```
while ( fscanf (fp, "%s_%s\n, old, new ) != EOF)
```

Hash table

- repeat the same process as adding word to bf.

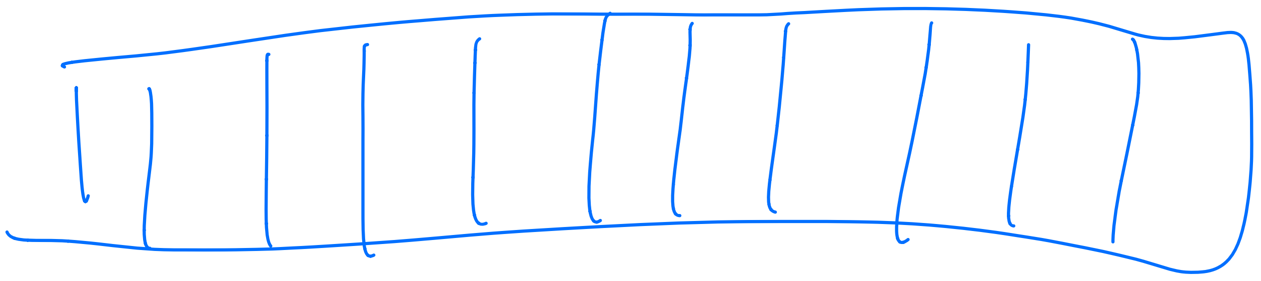
- but this time we also add hatterspeak in hatterspeak.txt.

Inside hashtable

- ht_insert

  ↳ ll_insert

ht_insert



- take the hashtable object, and the string

- hash the string
  - ↳ call ll_insert (ll, string)
    - ↳ according to the hash index, the ll_insert arrange the head of the linkedlist.

To user input:

1. Bloom filter check if the word is in bloom filter, probe().

   - if 1 of the 3 hash is not set, the word is filter.

2. Check hash table.
   go to heads[has].
   loop through the linked list.
   return NULL if not there.

3. Check if the word has hatterspeak.

- If Not. store the gs in "banned" Linked list.

- If yes, store the gs in "translat" Linked list.

⊗ If there is no hatterspeak, it means the word is not translable.

⊗ If the word is filtered by bf and ht, No output needed.

Output.

- display "banned"

- display "translate"

If -s flag set,

- output stat.