CS 636 Semester 2020-2021-II: Assignment 2

March 2021

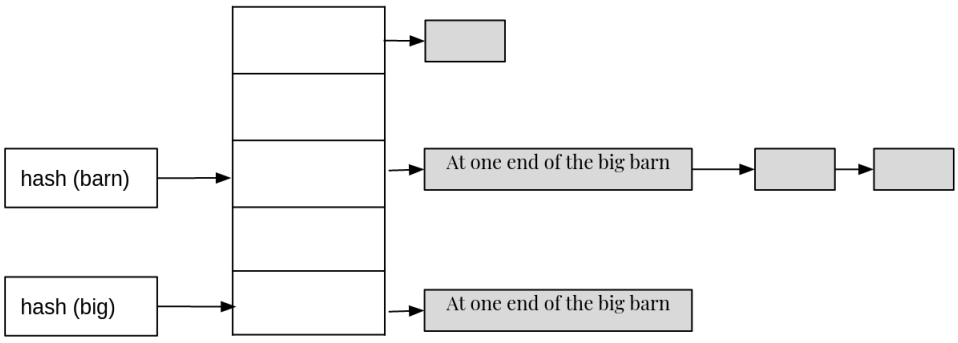
1 Introduction

You are required to implement a concurrent hash table to facilitate lookup of all sentences containing a given word. The input to your program will be a file containing list of comma separated words and a sentence per line as shown below.

Input format

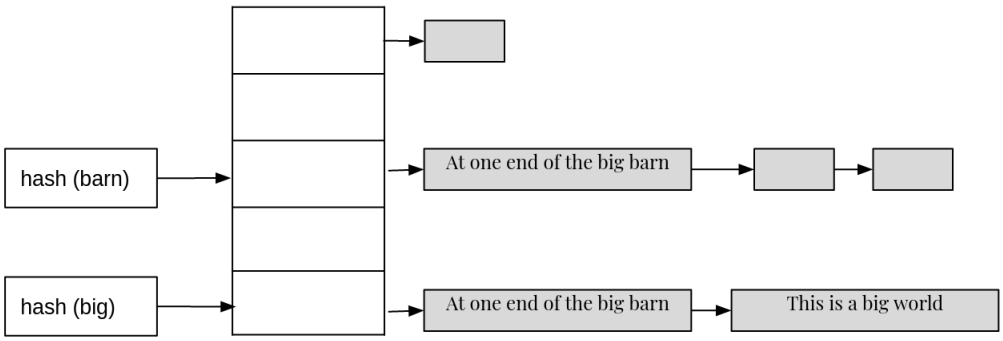
```
('one', 'end','big', 'barn') "At one end of the big barn"
('Major', 'already', 'ensconced','bed') "Major was already ensconced on his bed"
```

Your hash table should use `hash(word)` as key into the hash table and `sentence` as value as shown in figure **??**. The key will be of `MAX_KEY` and value will be of `MAX_VALUE` size as provided in the template code. The `sentence` should be inserted at `hash of all words` present in the list, refer to figure **??**. Your implementation should use chaining to resolve key collisions. The `hash()` function to generate key is provided in the template code.

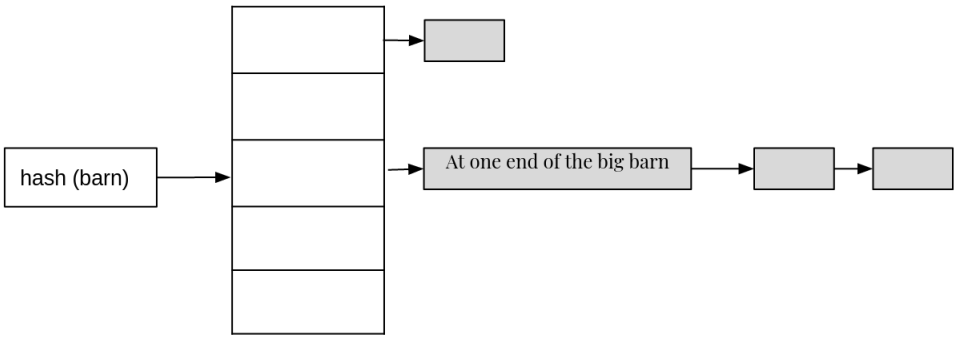


You are expected to provide following functionalities for the hash table.

- insert(word, sentence)
  Inserts the sentence at `hash(word)` locations in the table.
  insert("big","This is a big world")



- find(word)
  Returns all the sentences containing the word.
  `find(big)` returns
  "At one end of the big barn", "This is a big world"
- remove(word)
  Removes the word and associated sentence from the hash table.
  remove(big)



1.1 Part I

Implement blocking concurrent hash table data structure. You should use locking at the finest granularity such that operations to the same hash table location should only compete for acquiring a lock. Your implementation should not lock the entire hash table for operations, doing so will result in deduction of marks.

Size of the hash table is provided as `SIZE` in the template code. You don't have to handle hash table resize as part of this assignment.

`find(word)` should block if an insert of remove is going on for that word in the hash table.

1.2 Part II

Implement non blocking version of concurrent hash table in Part I. You should not use locks in this part of the implementation.

Note that lock free implementations are often vulnerable to the ABA problem. You can maintain a metadata information of count of sentences per word before and after to avoid ABA problem.

2 Test scenarios

- Your implementation will be tested for correctness by checking the state of the hash-table after performing concurrent `insert()`, `remove()` operations.