Hashing (II): Hashing with Chains

CSCD 300 – Data Structures

Eastern Washington University

EASTERN WASHINGTON UNIVERSITY
start something big

Goal

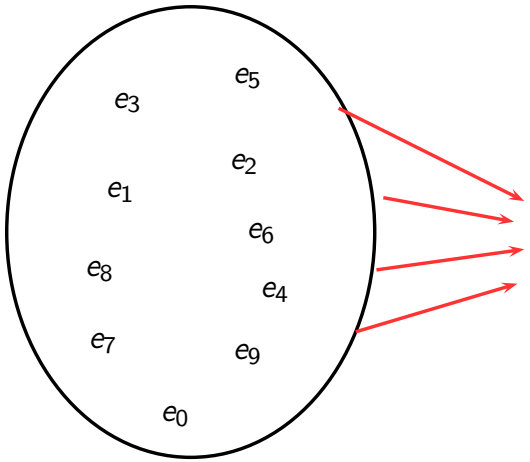
We will discuss the hashing mechanism that uses linked lists to solve the hash collisions. In the end, we will go over a few practical advices on picking a good hash function.

Outline

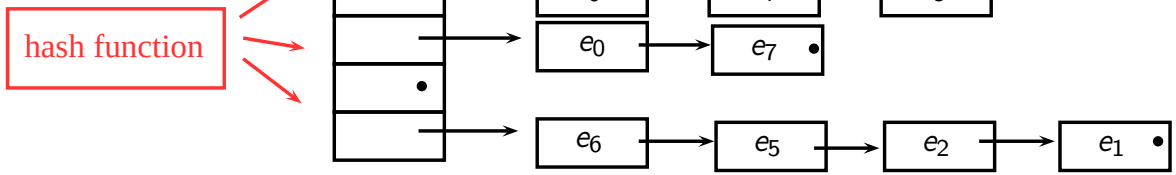
Hashing with chains

Key points

Allocate an array, where each array location is to save the reference to a linked list which will save all the elements that are hashed into that array location.



a data set of size 10



The whole hash table system with table size 5

There is no particular requirement on the order of the nodes in each linked list.

The structural design of the implementation

- Define and implement the linked list node class. The node needs to have the field(s) that represents the element. (We have discussed it when learning linked lists, but slight modification is needed.)
- Define and implement the singly linked list. (We have discussed it when learning linked lists, but slight modification is needed to support various hashing-relevant operations.)
- Define and implement the `HashChain` class that defines the hash table with chaining as follow ...

(continue ...)

The implementation pseudocode of the HashChain class [1]

```
interface map{
   int size(); /* return the number of elements in the hash table. */

   /* return the element whose id=key, if it exists;
      return null, otherwise */
   element get(key);

   /* if e.id exists, update that element's value to be e.v and
      return that element's old value; otherwise, insert e and return null. */
   value_type put(element e);

   /* Delete and return the element whose id=key, if it exists;
      return null, otherwise */
   element remove(key);  //delete an element with id=key

   /* any other methods that you need. */
   ...

}
```

[1] The code only presents the backbone structure and the main idea of the program's implementation. You need to understand it before you do the Java coding. Of course you can feel free to define your own interface and the specifications of the member API functions and the code structure.

```
class HashChain implements map{
   int size;
   SinglyLinkedList[] HashTable;

   HashChain(int table_size) {    //constructor
       size = 0;
       HashTable = new SingleLinkedList[table_size];
       for(i = 0; i < table_size; i++)
           HashTable[i] = new SinglyLinkedList();
   }

   int h(int key){   //hash function
      /* calculate and return the value of
         whatever hash function being used. */
   }

   int size(){
      return size;
   }

   (continue ...)
```

```
  element get(key){
     / * search in the corresponding linked list and return
         the result according to the API specification. */
    Node temp = HashTable[h(key)].search(key)

    if(temp == null) return null;
    else return temp.element;
  }

  value_type put(element e){
     / * search in the corresponding linked list and return
         the result according to the API specification. */
    Node temp = HashTable[h(e.id)].search(e.id)

    if(temp == null) {HashTable[h(e.id)].insert(e); size ++; return null;};
    else {old_value = temp.element.value; temp.element = e; return old_value;}
  }

  element remove(key){
     / * delete in the corresponding linked list and return
         the result according to the API specification. */
    Node temp = HashTable[h(key)].delete(key)

    if(temp == null) {return null;}
    else {size --; return temp.element;}
  }
  //any other methods that you need are here ...
}
```

A few advices on picking a good hash function

- The division method:

$$h(x) = x \mod N$$

  where N is the table size which is usually no smaller than the data set size. It is often good for "spreading" the elements if N is a prime number.

- The multiply-add-and-divide (or "MAD") method:

$$h(x) = ((ax + b) \mod p) \mod N$$

  where N is the table size which usually is a prime number and is no smaller than the data set size, p is a prime number larger than N, and a and b are integers randomly picked from $[0, p-1]$ with $a > 0$.