Documentatie Lab2 Week 3 LFTC

Socaciu Mihai 938

**Github link:** <https://github.com/Socca98/LFTC-2020>

Domain

Ht = {ht | ht is a hash table with elements of the type TElem}

Ex: {65: ['A'], 15: ['AT', 'TA']}

Interface for Hash Table

* init(ht)
  + **Description**: creates a new empty hash table (dictionary)
  + **Pre**: True
  + **Post**: ht Ht
* add(ht, e)
  + **Description**: hashes an element e for a key and links it to that key in the ht
  + **Pre**: ht Ht, e is a TElem
  + **Post**: ht’ Ht, ht’ = ht e
* get(ht, e)
  + **Description**: Returns the position of an element if exists. Position consists of index\_key in the dictionary and index in the list linked to that position.
  + **Pre**: ht Ht
  + **Post**: get <- (index\_key, cl\_index), e is a TElem

index\_key is the key integer after hashing

cl\_index is the index in the list linked on index\_key

* hash(ht, e)
  + **Description**: Hashes an element. Sum of ASCII characters % 67.
  + **Pre**: ht Ht, e is a TElem
  + **Post**: hash <- hash\_value

Representation

content: Dictionary{}

Dictionary

key: Integer

collision\_list: TElem[]

//Hashing function algorithm

**subalgorithm** hash(ht, e) **is**:

sums\_chars <- 0

**for** i <- 0, e.length **execute**

sums\_chars <- sums\_chars + (ASCII of e[i])

**end-for**

hash <- sums\_chars % 67

**end-subalgorithm**

Collision resolution

I chose **separate chaining**. Every key in the hash table has a list of elements where unique new elements with the same key are appended. 