Duy Do
Phung Tran
CPSC 335-03
Prof. Doina Bein
December 6, 2015

Programming Assignment 4: Cuckoo Hashing algorithm

1. **Pseudocode**

```
tablesize = 17      # cuckoo tables' size
t[tablesize][2][255] #combine the two 1D table into one 2D table

def place_in_hash_tables (string):
      placed = false                       #true means inserted successfully
      index = 0                            #index 0 is the first table, 1 is the second table
      counter = 0

      pos = get_hash_value(string, index)          #calculate the hash value for string

      #if not inserted and not out of range
      while((!placed ) && (counter < 2*tablesize)):
                  # if the hash value at index <pos> in the <index> hash table is available, place the string there
                  if t[pos][index] = 0
                              t[pos][index] = string
                              placed = true
                              return placed
                  # if the entry at index <pos> in the <index> hash table is not available
                  else:
                              #evict the old string
                              temp = t[pos][index]
                              #place the new string there
                              t[pos][index] = string
                              #place evicted string to the other table
                              if index == 0:
                                          index = 1
                              else:
                                          index = 0
                              string = temp
                              # find an available slot in the other table for the evicted string
                              pos = get_hash_value(string, index)
                              counter ++
      return placed
```

```python
# compute the hash value to find an available slot for a string
def get_hash_value(string, index):
    pos = 1
    #if the string is being inserted in the first table
    if index == 0:
            #find hash value of the first character of the string
            val = string[0] % tablesize
            #if string contains only 1 character
            if len(string) == 1:
                    return val;
            #if string contains > 1 characters, loop iterates i from 1 to the last element of the string
            for i in range(1, len(string)):
                    # find position
                    pos *= 37
                    pos = pos % tablesize
                    #find hash value of the whole string
                    val += string[i] * pos
                    val = val % tablesize
            return val
    #if the string is being inserted in the second table
    else:
            #find hash value of the last character of the string
            val = string[len(string)-1] % tablesize
            #if string contains only 1 character
            if (len(string) == 1):
                    return val
            #if string contains > 1 characters, loop iterates i from 1 to the last element of the string
            for i in range(1, len(string)) :
                    # find position
                    pos *= 37
                    pos = pos % tablesize
                    # find hash value of the whole string
                    val += s[len(string)-i-1] * pos
                    val = val % tablesize
            return val
```

2. **Table**

	Table T1	Table T2
[0]	One of the greatest	
[1]	Dynamic decision making	Server Problem
[2]	California	
[3]	Algorithm Engineering	Greatest mysteries
[4]	Quantum Nature of Universe	
[5]		mysteries in science
[6]	macroscopic quantum objects	emphasis on
[7]	In physics and	Self-Stabilization
[8]	Dynamic Programming	Optimal Tree Construction
[9]	Online algorithms	
[10]	Department of Computer Science	
[11]	String Matching	State University
[12]	Some related problem	
[13]	are known	astronomy
[14]		College of Engineering and Computer Science
[15]	Matrix Searching	Monge Properties
[16]	Fullerton	to scientists