Social Credit Score-

Bonta Sergiu Vlad - prelungire

[2018-06-07 Thu]

Contents

1 Problem statement

ADT SortedList – Implementation on a hash table, collision resolution by separate chaining.

By 2020, China plans to assign each of its 1.4 billion citizens a "social credit score" that will determine what people are allowed to do, and where they rank in society.

It's part of a broad effort in China to build a so-called reputation system that will measure, in theory, the credibility of government officials and businesses, in addition to citizens. The Chinese government says the system will boost "trust" nationwide and build a culture of "sincerity."

Since we are dealing with a huge data set we need fast inserts, fast lookups, and some way to keep storage space low. We also want to sort the data set according to each citizens score. For managing this huge undertaking I chose a Sorted list with a hash table as an implementation.

2 Implementation

2.1 Census

2.1.1 TEntry

field	field type
score	TInt
name	TString
email	TString

2.1.2 Repository

field	field type
ht	↑[THashTable]
relation	↑Relation

1. Operations:

    - init(ht, relation):
        - pre: relation is a relation
        - post: ht is an empty hash table
    - int compareTo(TEntry first, TEntry second):
        - pre: first, second Entry elements
        - post: compareTo returns:
            - -1 if the score of first is less than that of second;
            - 0 if the scores are equal;
            - 1 if score for first is greater
    - bool isFull():
        - pre:
        - post: return True if repository is full false otherwise
    - int length():
        - pre:
        - post: returns the number of entries in the repository
    - bool insert(TEntry entry):
        - pre: entry is a TComp
        - post: the element entry was inserted into the hash table to where it belongs

- Entry retrive(Tstring name):
  - pre: name is a string
  - post: returns the element with the name equal with name, error if not found
- void remove(string name):
  - pre: name is a string
  - post: remove the element with the name equal with name, error if not found
- void reset():
  - pre:
  - post: clears the entire hash table

2.1.3 Hash Table

field	field type
keys	[TInt]

1. Operations:
   - init():
     - pre:
     - post:
   - void insert(TEntry):
     - pre:
     - post:
   - TEntry retrive(TString name):
     - pre:
     - post:
     - post: removes the entry with the name equal with name
   - void remove(TString name):
     - pre: name is a TString
     - post: removes the entry with the name equal with name
   - clear:
     - pre:
     - post: clears the hash table

- isEmpty():
  - pre:
  - post: returns true if hash table if empty false otherwise
- bool isFull():
  - pre:
  - post: returns true if hash table if full false otherwise

2.1.4 HashTableIterator

field	field type
keys	[TInt]

1. Operations:
   - init():
     - pre:
     - post:
   - void insert(TEntry):
     - pre:
     - post:
   - TEntry retrive(TString name):
     - pre:
     - post:
     - post: removes the entry with the name equal with name
   - void remove(TString name):
     - pre: name is a TString
     - post: removes the entry with the name equal with name
   - clear:
     - pre:
     - post: clears the hash table
   - isEmpty():
     - pre:
     - post: returns true if hash table if empty false otherwise
   - bool isFull():
     - pre:
     - post: returns true if hash table if full false otherwise