Documentație Proiect Galben Casian-Petrică

Hash Table Hash Cache

1.1 Descrierea aplicației

Aplicația Hash CPU Cache implementează un nou tip de memorie cache, Hash Cache, care folosește tabele de dispersie pentru stocare.

1.2 Motivare

Am implementat această aplicație pentru a demonstra modul de funcționare al unui cache.

Ce este o memorie cache?

Este o colecție de date ce sunt o "copie la indigo" a valorilor originale stocate altundeva sau calculate mai devreme, unde operația de aducere din memorie a datelor originale este costisitoare sau costul recalculării acestora este mare, în comparație cu costul citirii acestora din cache. Cu alte cuvinte, un cache este o arie temporară de stocare unde datele utilizate în mod frecvent pot fi depozitate pentru un acces rapid la acestea. Odată ce datele sunt stocate în cache, în viitor vor fi luate de aici și utilizate decât să se încerce readucerea datelor originale sau recalcularea acestora, astfel încât timpul mediu de acces este mai mic.

1.3 Obiective

Exemplificarea modului de funcționare a unui cache și ințelegerea importanței tabelelor de dispersie, pentru eficiența lor.

2. Studiul actual in domeniul studiat

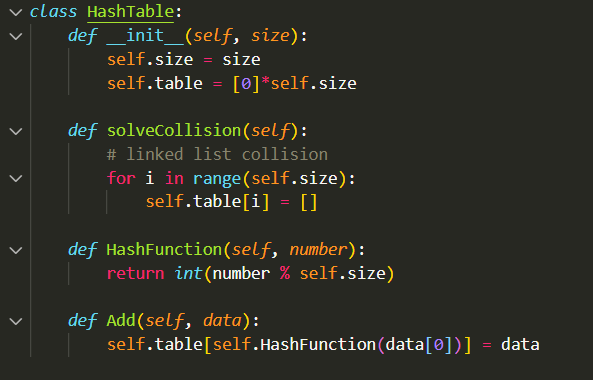
Tablelele de dispersie sunt un fel de mulțimi dinamice ale

cărora elemente sunt obiecte ale unui tip de date abstracte

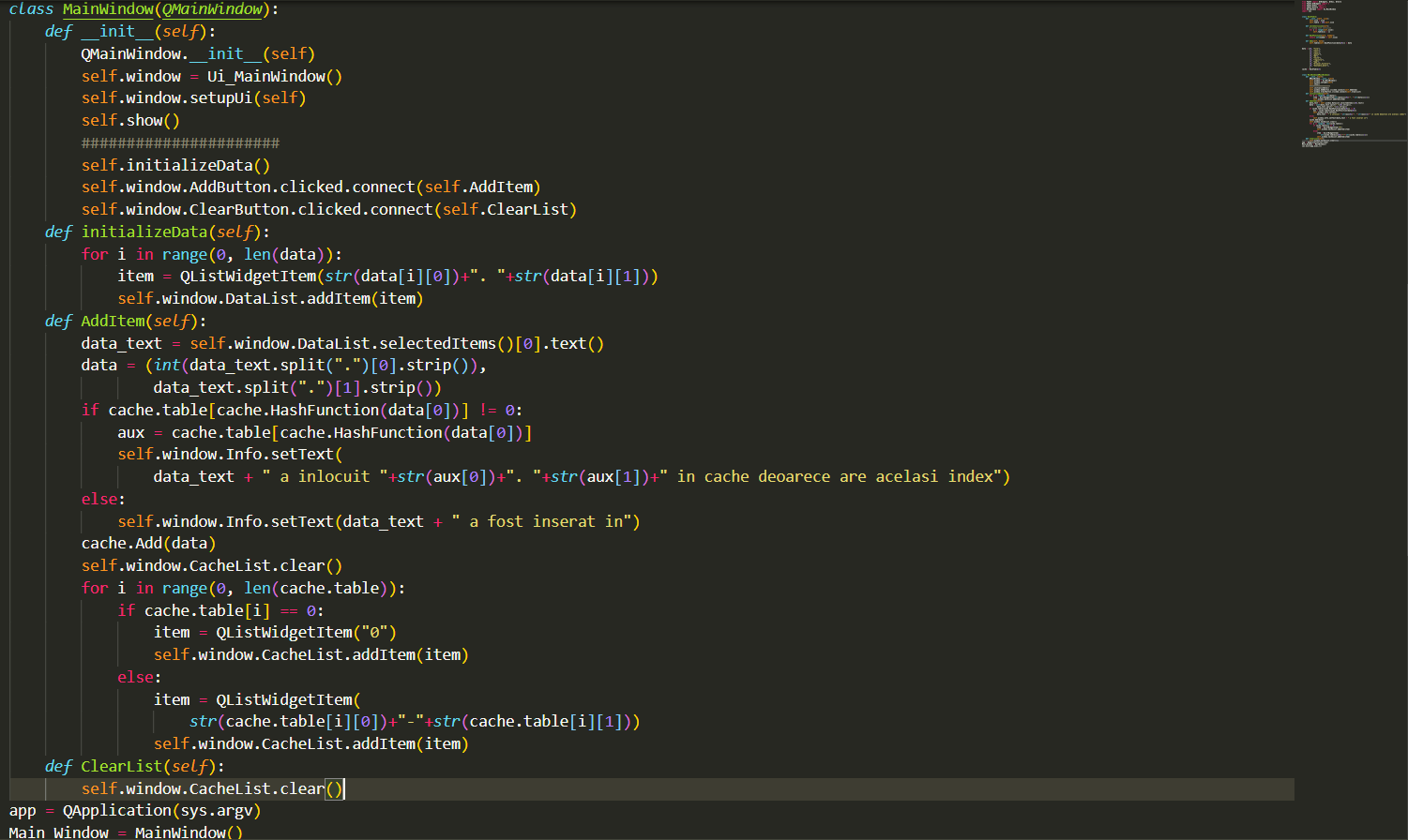
(ADT), cu un câmp cheie de identificare. Punctul forte al lor este eficiența, timpul pentru căutare, inserare și stergere fiind O(1).

Aceasta este implementarea HashTable in Python. Size reprezeinta dimensiunea tabloului(table), iar HashFunction() este o functie de dispersie. Add() adaugă un obiect în indexul calculat de funcția de dispersie.

Aplicația oferă 2 zone, un Main Memory și un Cache Memory. Diferența dintre cele două este că memoria cache este mult mai rapidă. Indexul datelor in memoria Cache va fii calculat din indexul lor in memoria Main, folosind o funcție de dispersie (restul împărțirii indexului din main la dimensiunea cache-ului). În cazul în care in memoria Cache există deja date intr-un anumit index, coliziunea va fii tratată prin înlocuirea datelor vechi cu cele noi.



Clasa MainWindow implementează interfața din clasa Ui\_MainWindow(), împreună cu funcționalitățile ei(butoane, text, liste, etc). Aici se face și implementarea adăugării în tabloul de dispersie(Cache), din Main Memory.



3.

Python 3.9

* Librăria PyQt5
* Qt Designer

