Aashray Shrestha

I do not give my instructor permission to share my answer with the class.

Report
CS 3310 : Data and File Structures
Assignment 3
Instructor : Dr.Ajay K Gupta
TA : Rajani Pingili

PHASE 1 : SPECIFICATION

1. The main goal of this assignment was to empirically and analytically compare different data hashing algorithms and the time complexity.
2. Implement a several hashing algorithms
3. Implement three probing algorithms (linear, pseudo-random, and double-hashing)
4. Practice good coding conventions

The application followed these guidelines:

1. Read and convert data from text file
2. Create n required bags with different hashing and probing algorithms
3. Fill each bag with 125 random elements from main data holder
4. Find random element in each bag
5. Write average time for each bag and each type of bag
6. Print internal data of bag and time values - raw data and average

PHASE 2 : DESIGN

1. Java Project
    a. App
        i. public App(Item[] itemsArray, int n) Construct application class and internal bags
        ii. public static void main(String[] args) Start point of program
        iii. private static Item[] readFile() Read data from file an return it as array
        iv. private void search(Item[] itemsArray) Search random value on bags and save time of each search
    b. Bag
        i. public Bag (Item[] items, int probingType, int numberBag, InitInterface init, HashInterface hash)Constructor. Create bag with selected type of hashing and probing, using defined interfaces
        ii. public String print(boolean fullType) Create text representation of bag
        iii. public int getNumber() Getter of bug number
        iv. public int[] find(Item item) Search Item in bug
        v. public int[] getStrengths(int[] res, Item item) Findlist of strengths of selected items.
    c. Item
        i. private Item() Private empty constructor
        ii. public Item(String line) Constructor, which create item, using string data

iii. public boolean equalsName(Item item) Check identity of name

d. HashInterface

i. int getHash(Item item) – create and return hash of item

ii. int getType() – return type of has

e. InitInterface

i. void init(Bag bag, Item[] items) – init bag with items

PHASE 3 : RISK ANALYSIS

No risk.

PHASE 4 : VERIFICATION

The algorithms were tested multiple times, and the searched items were printed out multiple times to verify that each step of the process was computed correctly.

PHASE 5 : CODING

The code is attached with this file and has comments to comprehend the code properly.

PHASE 6: TESTING

This application provides testing and comparison of three different types of probing data, using different algorithms to obtain a hash value.

The first research method is Open Hashing, in which an array of arrays is used as a hash table. Arrays help to avoid hash conflicts when different elements pretend to get the same array space in the hash table.

The simplest method for resolving hash conflicts is linear probing, in which the next free cell in the array is selected if the necessary cell is also occupied by another element.

A more complicated method is pseudo-random sounding, in which the transition to the next free place in the array is not linear, but pseudo-random (controlled), which, theoretically, helps to better fill the hash table.

There is a double hash probe that uses a second hash with a hash conflict. Count the total hash until I find free space

Space complexity analysis

Analysis of space complexity for the code is summarized in the table below:

Method	Input space	Private fields space
App	2	7 exemplars of created and defined interface objects, 3 helper methods and n*12 created bags
main	1	4
readFile	0	4+750 (total items created)

search	1	4+n*15 (StringBuilder objects and lists of find items)
Bag	5	8
print	1	1-2 (depending of type listing)
find	1	2
getStrengths	2	1
Item	1	5
getName	0	0
getMinimumStrenght	0	0
getCurrentStrenght	0	0
setCurrentStrenght	1	0
toString	0	1
equalsName	1	1

Time complexity

Table of time complexity of methods

	Worst-Case	Average-Case	Best-Case	Space Complexity
Open hashing	O(n)	O(log n)	O(1)	O(n)
Linear probing	O(n)	O(log n)	O(1)	O(n)
Pseudo random probing	O(n)	O(log n)	O(1)	O(n)
Double hashing probing	O(n)	O(log n)	O(1)	O(n)

Results

Empirical results are consistent with manual analysis, as shown below. Each data point is an average of 5 different instances of a random element search. Random numbers contained from 0 to m, where m is the maximum length of an array of elements.

The documentation of System.nanotime () shows that there may be some inaccuracies in the measured time, so one unregistered search circle provided during some warming of the Java virtual machine.

In all types of algorithms, the hash was calculated based on the sum of the name and rarity. due to the greater uniqueness of these parameters, and in accordance with the provision of a search on this data.

Each graph below shows the average search time for an item using a number of bags, such as hashing and verification. The Y axis is presented in log 10 for better readability.

First hashing algorithm sums and multiply values of bytes of hashed string. Mimics standard hash mechanism

Second algorithm was add and multiply char values of the string. Practically, that mechanism avoid hash collisions of different strings mostly.

Third algorithm sums up the action of previous hash algorithms.

In all variations except when n=1 time complexity is similar to theorized, O(log n).

PHASE 7 : REFINING THE PROGRAM

The program was refined along the way with changes in pseudocode that was written in the beginning.

PHASE 8 : PRODUCTION

Includes the source code and documentation

PHASE 9 : MAINTENANCE

Program is complete and maintenance/improvement can be done after the feedback is received.