Austin Frey

Professor Ling

CS-300

11/20/2022

Module 4 Hash Table Pseudocode & Reflection

**Reflection**

In this module’s assignment, I was tasked with developing code for a hash table implementation of the auction bids program that has served as a practice template throughout this class. Creating a hash table expedites search functions by using hash equations and keys. I initially encountered a lot of difficulty during this assignment because there was a lot of pseudocode in the Zybooks lessons. Still, some functions and techniques weren’t covered in much detail. However, that just meant I needed to review the Zybooks lessons and do additional research to understand hash tables. Finally, after browsing various stack exchange posts, I understood how to implement hash tables much better.

**HashTable Pseudocode**

*Initialize structures to hold bids:*

Function HashTable()

Pass in: None

Resize hash table to default size

Pass out: None

Function HashTable(unsigned int size)

Pass in: size of the hash table being created

SET tableSize equal to size

Resize hash table with tableSize

Pass out: None

*Free storage when a class is destroyed*

Function ~HashTable()

Pass in: None

Create empty vector and swap current hash table for empty vector

Pass out: None

*Calculate hash value using bid ID*

Function hash(int key)

Pass in: the value of the key to be calculated

RETURN key modulo tableSize

Pass out: hash value

*Insert a bid*

Function Insert(Bid bid)

Pass in: Bid to insert

CREATE hash key from bidId

CREATE new local node called node and set to the node at hash key index

IF node is null

ASSIGN a new node with bid and the hash key

ELSE IF node’s key is the max number

ASSIGN node with the values of the hash key and the bid

SET node next pointer to null

ELSE

WHILE node->next points to another node

SET node to node->next

END WHILE

SET node’s next node to a new node with the hash key and bid

END IF ELSE

Pass out: None

End function

*Print all bids*

Function printAll()

Pass in: None

FOR all nodes in hash table

IF currentNode key is not the UNIT\_MAX

PRINT key and bid information

SET newNode to currentNode’s next pointer

WHILE newNode is not null

PRINT key and bid information

SET newNode to newNode’s next node

END WHILE

END IF

END FOR

Pass out: None

End function

*Remove a bid*

Function remove(string bidId)

Pass in: bid id to remove

CREATE key from bid id

CREATE node from key

CREATE bid from node’s bid

IF node’s bid id equal to bidId and node’s next pointer is null

DELETE the node at the key index

RETURN

END IF

IF node’s bid id equal to bidId and node’s next pointer is not null

CREATE tempNode set to node’s next pointer

DELETE node at hash key index

INSERT the temporary node

RETURN

END IF

IF node’s bid id doesn’t match and node’s next pointer is not null

CREATE previousNode from node

SET node to node’s next pointer

WHILE node is not null

SET bid to node’s bid

IF bid’s bid ID matches bidId

SET previousNode’s next pointer to node’s next pointer

DELETE node

RETURN

END IF

SET previous node to node

SET node to node’s next pointer

END WHILE

END IF

Pass out: None

End Function

*Search for a bid*

Function Search(string bidId)

Pass in: bidId to search for

CREATE bid object

CREATE hash key from bidId

CREATE node from the node associated with the hash key

IF node’s bid ID matches bidId and node’s key is not UINT\_MAX

RETURN node’s bid

END IF

IF node’s key is not UINT\_MAX and node is null

RETURN empty bid

END IF

WHILE node is not null

IF node’s bid ID matches bidId and node’s key is not UINT\_MAX

RETURN node’s bid

END IF

SET node to node’s next pointer

END WHILE

RETURN bid

Pass out: Found or empty bid

End function