Daniel Schween

CS 300 SNHU

07/27/2024

**Module Four Assignment Overview and Pseudocode**

This assignment uses hash tables to read, point and iterate through a file of bids. The bids are stored in a vector where a series of node commands is used to navigate through. “Node”, “currNode”, “tempNode”, “cursor”, “null”, and “null pointer” are some of the common variables declared and used to move through the list of bids. Head node and tail node are utilized to help sort through a list reorganizing the bids based off what the function is being used for.

“bidId” helps point at a particular bid and assign node variables to search through the list of bids. Nested loops are a major part of the assignment and “newNode” and “bid.next” help search through the vector one placement value at a time. Head/Tail nodes are used to reorganize the list of bids moving a particular value to the top of the list or the bottom of the list. Delete, remove, search, size, etc. were also methods used to manipulate the hash table, depending on the functions output. Hash tables utilize a key and a value which simplifies the list to navigate through the list and implement different functions.

PSEUDOCODE:

**INITIALIZE** Node and Bid with assigned keys

**INITIALIZE**  bid, key, node, and size of vector to default\_size

**SET** HashTable size to setSize

**RETURN** key modulus with the setSize

**CALCULATE** key and bid string to ensure key = bid

**INSERT(BID):**

**CALL**  hash() with bidId and store in tempKey

**IF**  hash location is empty

**CREATE** newNode with bid and tempKey

**SET** hash location equal to newNode

**IF** has location not empty

**CREATE**  a new node pointer with bid and tempKey

**LOOP** until the end of linked chain and add newNode

**END**

**SEARCH(STRING):**

**CREATE** new node pointer called cursor

**SET** cursor to the bucket at the hash location

**LOOP** until cursor is NULL

**IF** node at cursor contains a bid equal to string

**RETURN** cursor

**SET** cursor equal to nextNode

**END**

**REMOVE(STRING):**

**SET**  cursor to the bucket at the hash location

**CREATE** a newNode pointer called tempNode

**IF** a chain

**CHECK** if bidId matches string

**IF** match **SET** tempNode to the next node

**SET** cursor equal to tempNode

**DELETE** tempNode

**IF** not a chain

**SCAN** list for match

**IF** single bucket

**SET** bid members equal to the default constructor

**END**

**METHODS:**

**IF** node is “NULL”

**THEN** node = new node

New Node = bidId and tempKey

**ELSE**

**WHILE** node != null

Node = next

**FOR** node.find = node start

Node.find != node end

**WHILE** node != null

**PRINT**  bid

**IF** key = hash && bidId

Cleanup memory

**IF** node != null && tempKey = max

**RETURN**  node to bid

**WHILE** node != max and bidId = 0

**RETURN** bid

**MAIN():**

**READ** cmd arguments

**STORE** argument as CSV file path

**IF** no cmd arguments load default CSV file path

**LOOP** while choice is not equal to ‘9’

**OUTPUT** menu

**GET** user input; Store in choice

**VALIDATE** user input

**IF** choice is not 1-4 or 9 throw an error

**IF** choice equals ‘1’

**START** the clock and store in ticks

**CALL** loadBids and store CSV data in HashTable *bidTable*

**OUTPUT** number of records in the CSV file

**STOP** the clock

**OUTPUT** the elapsed time needed to read in the CSV file

**IF** choice equals ‘2’

**CALL** PintAll() with *bidTable*

**IF** choice equals ‘3’

**START** the clock and store in ticks

**CALL** Search() passing a *bidKey* to search for

**STOP** the clock

**OUTPUT** the elapsed time needed to find the *bidKey*

**IF** choice equals ‘4’

**CALL** Remove() passing *bidKey*

**IF** Choice equals ‘9’

**EXIT** the application

**OUTPUT** ‘Good bye’

**END**