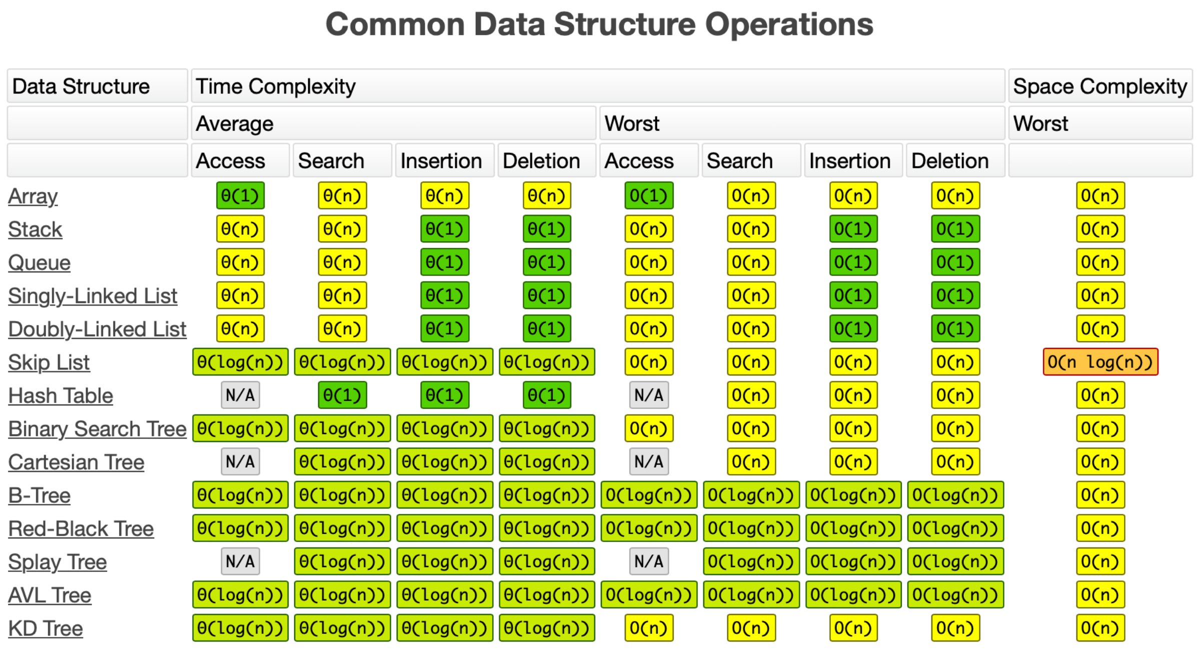
Aditya Lamba

PROG20799

Professor Tiffany Antopolski

August 6, 2024

Assignment 3 – Big O Analysis



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

makeNode (int n):

- This function is inserting a node, and using the chart for a singly-linked list the big O value for this is O(1).

insertNodeAtHead(NodePtr head, int n):

- This function is also inserting a new node at the beginning of the list according to the chart the big O for this function is O(1).

listLength(NodePtr head):

- This function is searching through the list to gauge the length. According to the chart it has the big O(n).

listLength_recursive(NodePtr head):

- This function is also searching through the list to gauge the length. According to the chart it has a value of O(n).

listSum(NodePtr head):

- This function must access the value of each node in order to add them up to find the sum. According to the chart singly-linked list access has a big O(n).

listSum_recursive(NodePtr head):

- This function similarly uses access to each node's value. It is big O(n).

printList(NodePtr head):

- This node requires access to the value in each node in order to print. It has a big O(n).

printList_recursive(NodePtr head):

- This node requires access to the value in each node in order to print. It has a big O(n).

printListReverse(NodePtr head):

- This node requires access to the value in each node in order to print in reverse. It has a big O(n).

printListReverse_recursive(NodePtr head):

- This node requires access to the value in each node in order to print in reverse. It has a big O(n).

The approach taken with this assignment was different for each function. When required to use an iterative function, then the approach taken was to use loops to effectively move through the list.

When the function required a recursive function, then the approach was to print at the current source, move to the next location pointed at and repeat.

Challenges I faced were staying within the c90 parameters, making sure int i was declared and initialized before being used in a loop. I also had issues when returning 0 in a void function since that was not possible and I just had to do return;

I had to remember to free the temporary list that was created in the end of the function so that there were no memory leaks, made it so that it would not have to be done in the main.

I could have made more error handling malloc so that in case the memory was null I could have returned a message.

I had to specifically remember to free the list and create a function to do so since it was not in the instructions.