Mohit Sharma (2019CS10372)

A3.2: Dynamic memory allocator using BSTree.

In a BSTree of n nodes and h height.

Worst case time complexities of some functions are as follows:

Insert : $O(h) \Rightarrow O(n)$

Delete : $O(h) \Rightarrow O(n)$

Find : $O(h) \Rightarrow O(n)$

In worst case height of BSTree can be n.

After n operations in Dynamic memory Allocator:

Worst case time complexity of Allocate : $O(n)$.

In allocate function, a find, insert and delete is called on freeBlk and a insert on allocBlk.

$T(n) = O(h) + O(h) + O(h) + O(h) = O(h)$

In worst case $T(n) = O(n)$.

The worst case can occur when n operations of allocate are made, so by doing this there will be n nodes in allocBlk ~~as~~ with its height $= n$ with every node having only right child (other than leaf node).

This will give $T(n) = \underbrace{O(1) + O(1) + O(1)}_{\text{Find, delete \& insert in freeBlk will be } O(1) \text{ as it has only one node in the scenerio made.}} + O(n) = O(n)$

Worst case time complexity of Free : $O(n)$.

In free function, find & delete is called on allocBlk and insert on freeBlk.

$T(n) = O(h) + O(h) + O(h) = O(h)$

In worst case $T(n) = O(n)$.

With same example as in allocate, if free function is called with largest address allocated than

$T(n) = \underbrace{O(n) + O(n)}_{\text{Find \& Delete in allocBlk}} + O(1) = O(n)$

Mohit Sharma (2019CS10372)

Worst case time complexity of Defragment : $O(n^2)$

In defragment function we make a address based free memory block (i.e, address as key) by traversing the original freeBlk.

The traversal using getNext function has a amortised time complexity of $O(n)$ as every node is visited only two times in ~~the~~ all the calls of getNext function.

At every node, we make a copy of the values of that node inserted in the address based FMB.

So, $T(n) = \underbrace{n}_{n \text{ nodes}} \times \underbrace{O(n)}_{\text{Worst case time complexity of insertion.}} = O(n^2)$ [Actually it is like $T(n) = 1 + 2 + \dots n = O(n^2)$]

Then we traverse the address based FMB and deletes the corresponding continuous ~~blks~~ blocks from original FMB, and insert a new block in place of them.

This is again $O(n^2)$ in worst case as all the nodes can be deleted from original FMB, so n ~~nodes~~ calls to deletion function gives:

$$T(n) = n + n-1 + n-2 + \dots + 1 = O(n^2)$$

During first deletion there are n nodes in FMB, during second deletion there are n-1 and so on.

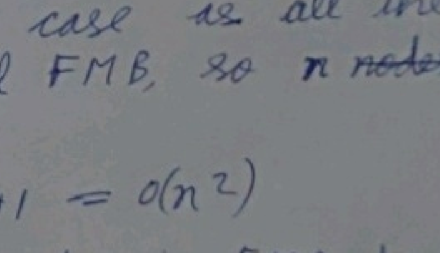
So worst case time complexity is $O(n^2)$.

This can occur when n operations are done in following way:

(i) $n/2$ operation of allocate

(ii) $n/2$ operation of free with smaller address block freed first.

This will create ~~first~~ freeBlk as : →



● $n/2$ nodes & $O(n)$ height ($n/2$ height)

While creating address based FMB, a similar BST will be formed in $O(n^2)$ time complexity because at i th iteration of traversal, i-1 nodes will be there in address based FMB, so $T(n) = \sum_{i=1}^{n/2} (i-1) = O(n^2)$.

Mohit Sharma (2019CS10372)

A3.3: Dynamic memory allocator using AVL Tree.

In a AVL Tree of n nodes and h height

Worst case time complexities of some functions are as follows:

Insert : $O(h) \Rightarrow O(\log n)$

Delete : $O(h) \Rightarrow O(\log n)$

Find : $O(h) \Rightarrow \cancel{O(n)}\ O(\log n)$

In worst case, height of AVL Tree is of $O(\log n)$.

After n operations in Dynamic memory Allocator:

Worst case time complexity of Allocate : $O(\log n)$.

In Allocate function, find, delete & insert are used which have worst case time complexity of $O(\log n)$

Worst case time complexity of Free : $O(\log n)$.

In free function also, find, delete & insert are used which have worst case time complexity of $O(\log n)$.

Worst case time complexity of Defragment : $O(n \log n)$.

In defragment the traversal done on FMB takes $O(n)$ times as getNext function goes through all nodes two times.

In the while loop of traversal we insert a node in address based FMB.

So, $T(n) = 1 + \log 2 + \log 3 + \ldots \log n = \log n! = O(n \log n)$.

then we traverse through address based FMB and delete corresponding contiguous blocks from original FMB, we may need to delete all nodes in worst case.

$T(n) = \log n + \log n-1 + \ldots \log 2 + 1 = O(n \log n)$.

Same cases can be taken to analyse worst case time complexity as taken in A3.2.