COL106: Assignment 3

Sayam Sethi (2019CS10399)

November 2020

1 A3.2

Using the same argument as presented in A1.3, after n operations the size of `freeBlk` and `allocBlk` will be $O(n)$ and height of both will be h which is $O(n)$ for `BSTree`.

1.1 `Allocate`

`Allocate` calls `Find`, `Insert` and `Delete` on blocks of size of $O(n)$. The complexity of each of them is $O(h)$ for `BSTree`. Therefore the total complexity of `Allocate` is $O(h) = O(n)$.

1.2 `Free`

`Free` calls `Find`, `Insert` and `Delete` on blocks of size of $O(n)$. The complexity of each of them is $O(h)$ for a BSTree . Therefore the total complexity of `Free` is $O(h) = O(n)$.

1.3 `Defragment`

Initially, `Defragment` builds a new `Dictionary`. This takes $O(n * h_{addrFreeBlk})$ time (expected time is $O(n \log n)$). This is because traversal of `freeBlk` takes $O(n)$ time and insertions in `addrFreeBlk` takes $\sum_{i=1}^{n} O(h_i)$ which is bounded by $\sum_{i=1}^{n} O(h_{addrFreeBlk}) = O(n * h_{addrFreeBlk})$. Here, in the worst case analysis, $O(h_{addrFreeBlk}) = O(h)$. Additionally, for `BSTree`, $h = O(n)$.

The next step in `Defragment` traverses `addrFreeBlk` and calls `Delete` and `Insert` on it and on `freeBlk`. This loop runs in $O(n * h)$ time since `Delete` and `Insert` are $O(h)$ and the loop runs n times.

Deletion of all nodes from `addrFreeBlk` also takes $O(n * h)$ time and hence the total complexity of `Defragment` is $O(n * h) = O(n^2)$.

2 A3.3

The arguments as above are valid even for `AVLTree`. The only difference is that $h = O(\log n)$.

2.1 `Allocate`

The time complexity of `Allocate` is $O(h) = O(\log n)$.

2.2 `Free`

The time complexity of `Free` is $O(h) = O(\log n)$.

2.3 `Defragment`

The time complexity of `Defragment` is $O(n * h) = O(n \log n)$.