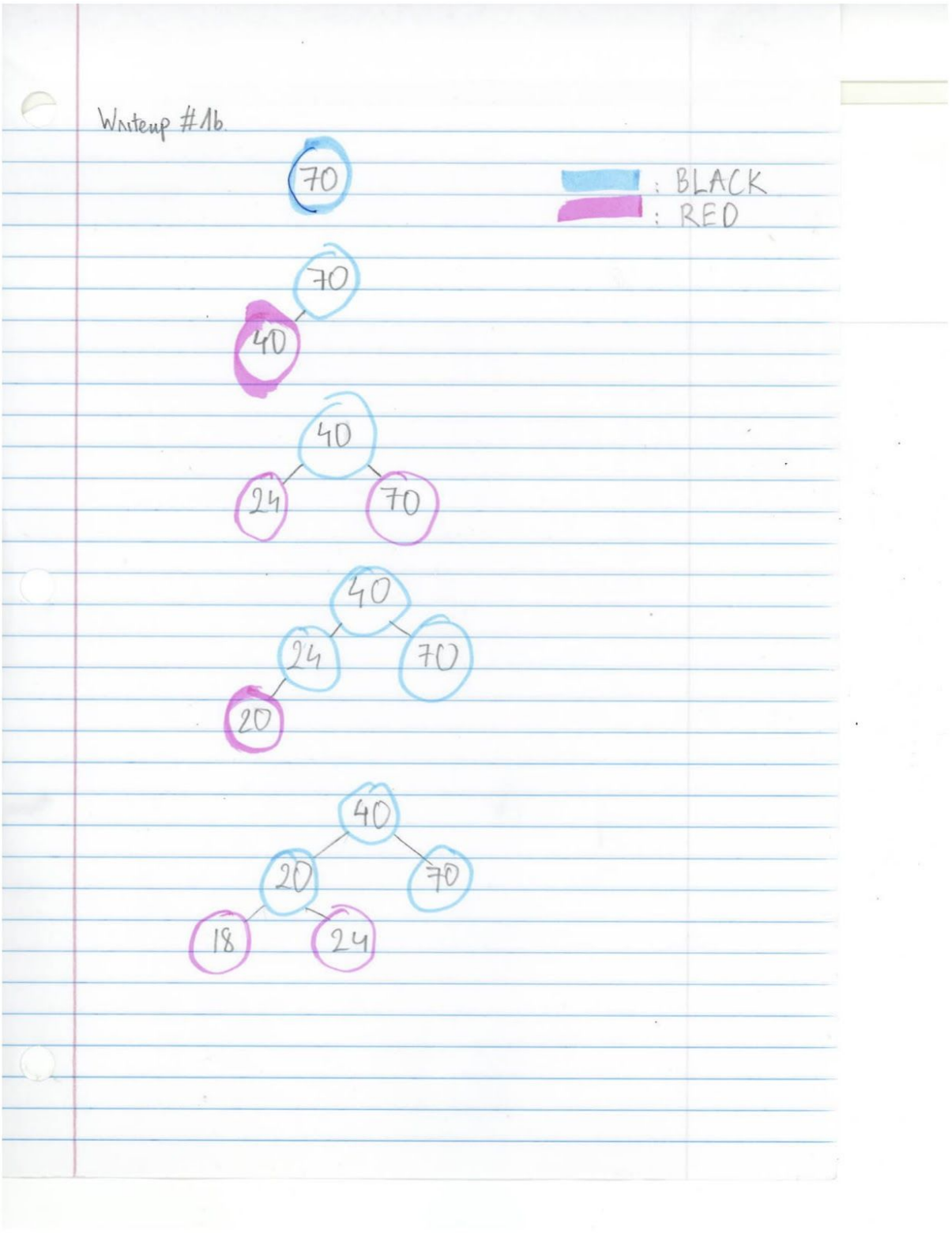
Lam Dang
CS335: Red Black Tree
Writeup

Writeup #1a:
Red-Black Tree is a self-balancing Binary Search Tree that has an extra feature: Color. These Color will ensure that the tree is approximately balanced during insertion and deletion. The Color rules needed to ensure that Red Black Tree is balanced are:
1/ Each Node is either Red or Black.
2/ The root is Black
3/ If a node is red, both of its children are black.
4/ Every path from a given node to a descendent Null Node have the same number of black nodes.

Writeup #1b:



Writeup #1c:

Red-Black Tree and (2,4) Tree are secretly the same thing because:

- All operation have the time complexity of O(logN)
- (2,4) Tree insertion and deletion cause nodes to expand, split and merge, which are conceptually similar to the Color Switching and Rotations that Occur in Red-Black Tree

Writeup #2:
The use case I'm using is Thread Scheduling, when you have 150 threads ready to run a job. For every job that I need to run, I go out to find an available thread and I run it. In my case, I use all of 150 threads using insertion, look up and delete the thread when it is done. We can see from the use case, my insertion and lookup in Red Black Tree are very fast, comparing to thread scheduling in Binary Search Tree. Because Thread Scheduling has a tendency to grab the thread in a sorted manner, Insertion and lookup of in BST cost O(logN) time when in RBT, it is self-balance, therefore it is easier for Insertion.and lookup. Deletion have worse time because of all the rotation and recolor. Therefore, in the Thread Scheduling case where inserting jobs into threads is abundant, I choose RBT over BST.

Writeup #3:
One of the reasons why Red-Black Tree is more favored that the AVL Tree or SplayTree is that Red-Black Tree can be implemented to do insertions using a loop rather than an insertion. In some machines, Recursion can be very expensive if you overrun function call cache. That is why Red-Black Tree is more favored than AVL in tasks such as kernel scheduling.

Another reason is that Red-Black Tree insertion and deletion is much faster than AVL Tree. Therefore, when implementing libraries such as map and multimap where insertion and deletion operations are more likely than the search operation, which AVL is good at.