Project Report

# • Overall Status

The *merge*

function was implemented successfully, implementing the delete algorithm for B+ Tree. I used the redistribution and merge concept. The Occupancy factor in the page is the most important factor to determine whether we redistribute or merge or just delete the key without any change to the B+ tree structure. When the condition falls below 50%, we merge the page with its sibling. Removing or keeping the levels of B+ tree indexes. We restore to redistribution most of the times by calling the redistribution function already provided.

# • File Descriptions

**Data structures:**

Stack:

I created a data structure for stack in order to store the information about the path information to the entry that is being deleted. I am using iteration in order to delete the entries in the leaf pages of the tree. Redistribution and merging at the leaf level of the tree is handled in *\_delete* function itself. But when we need to rebalance the tree, we need information about the path that has been traversed that leads to the deletion leaf page. This stack data structure is hence used in order to maintain this information.

**Function:**

# Merge:

This is a recursive function which handles the merge part of the deletion operation. We use this function when a merge has happened at the leaf level and may lead to the tree being modified.

# • Division of Labor

This project needed around 150 hours of time to write the code and the project report. It’s a team of one and hence division of labor is not applicable.

# • Logical errors and how you handled them

Logical errors:

1. Handling of pinning and unpinning of pages: In case of pinning and unpinning of pages, the recursive function merge had to pin and unpin a few pages while it calling function *\_delete* also had to unpin and free a few pages. This process of division of pinning and unpinning of pages was not handled gracefully. I fixed this by making the *\_delete* and *merge* unpin pages that they have pinned and hence solving the pin and unpin problem.
2. Merge code: Merge as a recursive function has details that need to be passed to the next function. Handling this was not done very well initially and later I fixed this using the stack that I created in order to pass information between the functions.
3. Finding the right and left sibling: I used the stack to save information about the path. But finding the right and left sibling was a problem and hence I stored the right and left sibling in the stack along with the path information.
4. Collapsing the level of the tree in case of merge: Initially iteration was used in order to implement merge. But, collapse of the level of the tree was not happening correctly. Hence, the recursive function was implemented in order to handle this situation.