5.2 Introduction to Super-B-Trees

A detailed description of the Super-B-trees will be deferred until Chapter 6 of this thesis.

This section will provide a brief introduction to Super B-trees. It will also explain the ~~[illegible]~~ of ~~[illegible]~~ structures. I will explain how the Super B-trees solve a very important ~~[illegible]~~

combinatorics problem.

The Super B-trees are a generalization of the traditional notion of "B-Trees." They will possess all the characteristics that have been traditionally assigned to B-trees plus ~~of leaves in Super-B-trees will thus be constitute a directory where the~~

the properties as the common characteristics that will be shown to be shared between traditional B-trees and Super-B-Trees are listed below:

1) both will possess $\Theta(\log N)$ heights
2) both will support $\Theta(\log N)$ insertion, deletion, and search operations
3) both will possess a data-structures where the leaves are arranged in order by

by increasing Key-value.

The distinction between Super-B-Trees and the traditional notion of B-Trees can be best explained if the interior nodes of these trees are examined in further detail. Note that B-trees had a very simple interior node structure. These nodes thus merely contained that information

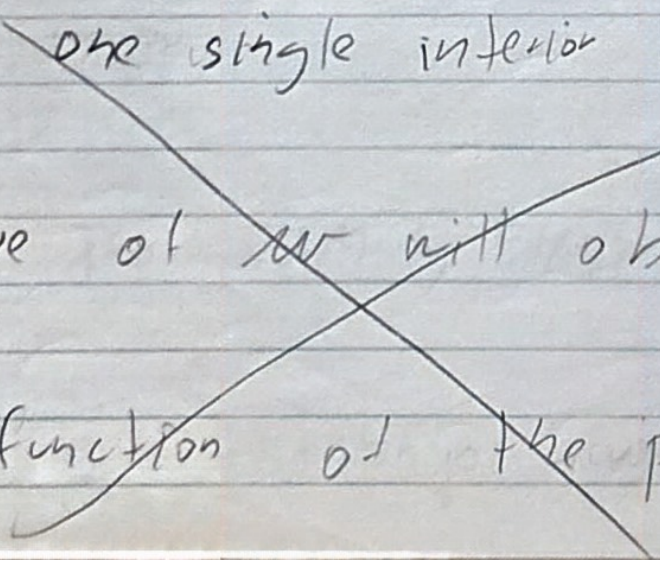
The SDS will play a crucial role in this chapter.

The entire databases of the SUM-3, FIND-3 and COUNT-3 algorithms frequently will be stored in the SDS structure.

This method will enable many algorithms to acheive a significant savings in runtime.

The Super-B-tree algorithms ~~will manipulate the SDS structures~~

will use a large number of different SDS structures in this thesis. Throughout this thesis, the symbol "w" will denote the worst-case time required to insert or delete a given y-record in the SDS-structure. The value of w will obviously be a function of the particular



value of w will obviously be a function of which SDS structure is employed. For example, the previous two chapters showed that the SUM-3 ~~will have an O(1)~~ ~~magnitude when SDS represents~~ ~~the SUM-3~~ that the SUM-3, FIND-3, and COUNT-3 databases would all

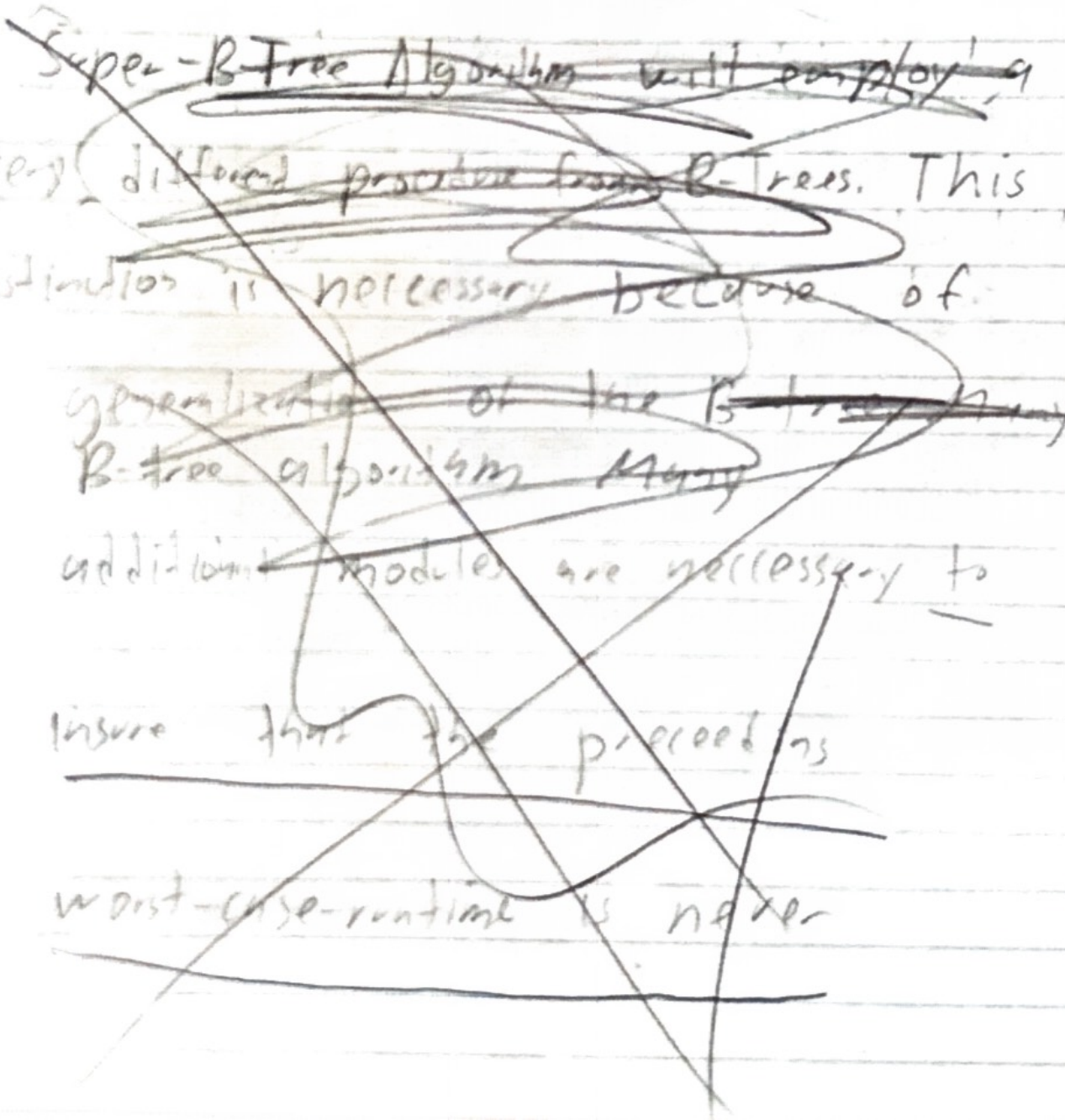
support $O(1)$ data-modification operations. Consequently, we will have an $O(1)$ magnitude when the SDS is any one of these three structures.

The main theorem in chapter 6 will state that any y-leaf can be inserted into

from a Super-B-tree

$O(\log N)$ WORST-CASE-TIME.

The Super-B-Tree Algorithm will employ a very different process from B-Trees. This distinction is necessary because of the generalization of the B-tree algorithm. Many additional modules are necessary to insure that the proceedings worst-case-runtime is never



proof of our Super B-Tree

Theorem and the procedure employed by it

~~by the different algorithm we~~

are very different from their traditional B-Tree

counterparts. The reason for this

distinction can be understood if traditional

B-trees are examined in further detail.

~~$O(\log n)$ worst case runtime is~~

~~re-examined.~~

Traditional B-tree will not operate efficiently if an SDS field is added to them. This ~~an addition of an SDS field will cause the worst~~ field will cause the worst-case cost of B-tree insertion and deletion operations to rise above $O(w \log N)$ wtime.

The reason for these difficulties can be understood

the SDS field in such

worst-case insertion and deletion time.

The difficulties with B-trees can

~~to start~~ if the case is

considered where the modification of

one or two points causes N/4

leaves to gain a new ancestor. In this

case, expensive N/4 time would

be needed to modify the

corresponding SDS structure of traditional B-trees. These B-trees are thus unable to only employ SDS structures because they too frequently engage in ~~pointer changing operations~~. such pointer-changing operations. Only the Super-B-Tree algorithm  will be shown to insure $O(w \log N)$ runtime even in

worst-case circumstances.

The Super-B-Tree Theorem

will be proven in Chapter 6. The

~~Chapter will use that theorem,~~

correctness of that theorem will

be assumed in this chapter. Our

various database algorithms will thus

be based on the presumption that

Super-B-Trees operate correctly.

I recommend that the

~~reader complete this chapter before~~

~~examining Chapter 6.~~

reader accept this assumption and that he should read chapter 5 before 6. It is not necessary for the reader to follow this course. Chapter 6 and its Super-B-Trees can be examined immediately.