Student ID: M00674847

Report For Coursework 2

**Introduction**

This is a project where I try to design a fast and efficient library system with an attempt at the hardest Red and Black tree which then turned into hashing with separate chaining as I literally converted some code from red and black tree that I tried to make.

The report will be as follows:

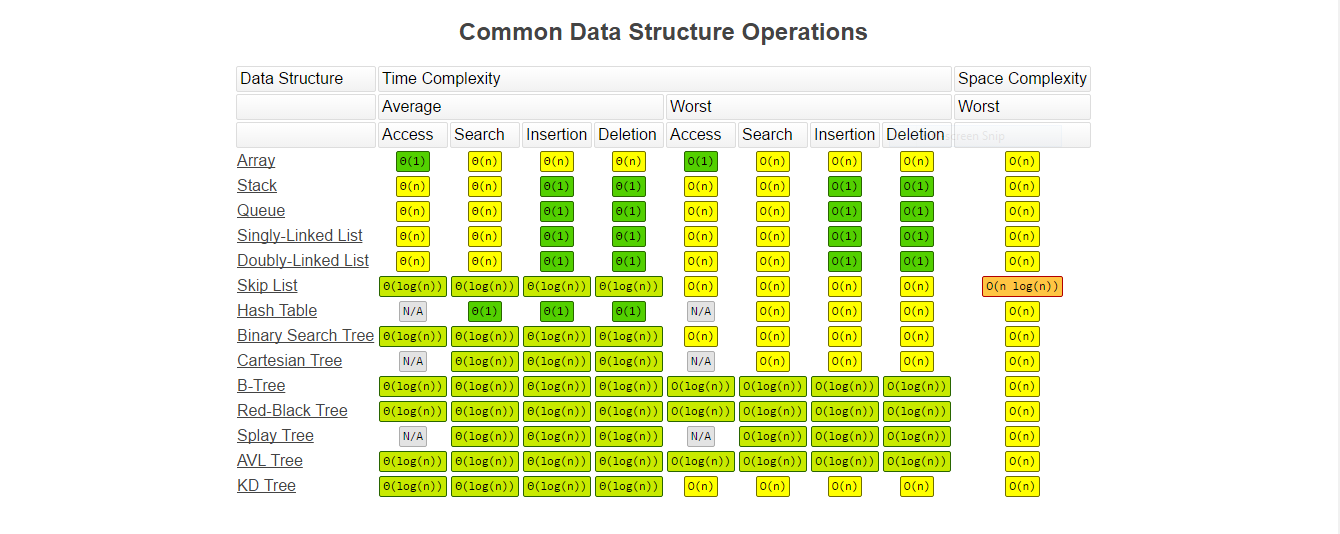
* Into.
* Git
* Design Choices
* Testing
* Conclusion
* Ref

**Git**



My git is pretty straightforward as you can see I had a few commits which first two or so of were me attempting a red and black tree. Bitbucket helped me correct some errors at times which was useful.

**Design**



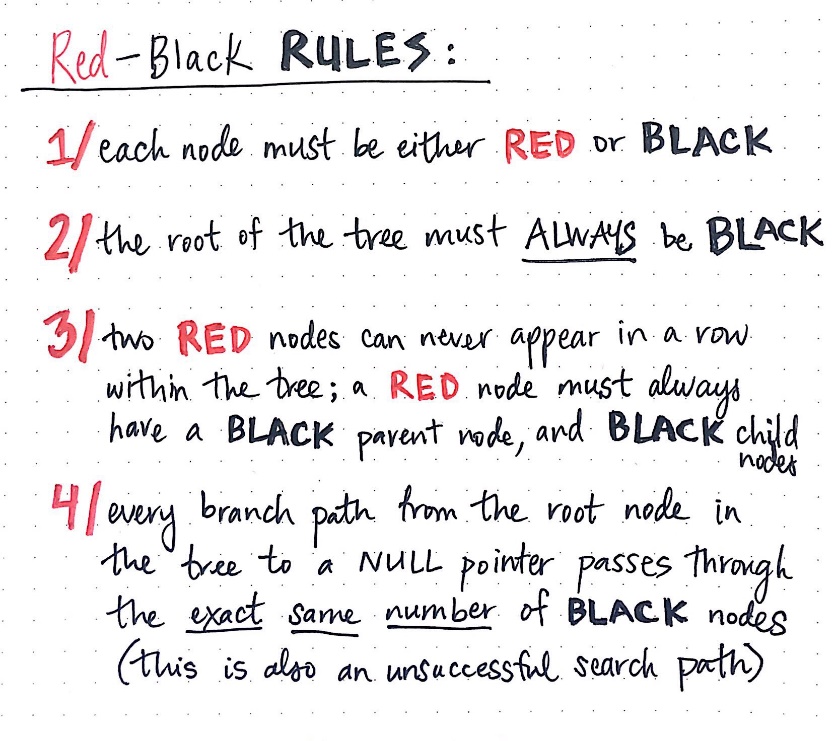
The goal was to pick a fast and efficient algorithm to find books and allocate them efficiently. Red and black tree was the most efficient as it can balance it self with additional rules meaning there is never a situation where you are stuck with a long chain.

Linked lists:

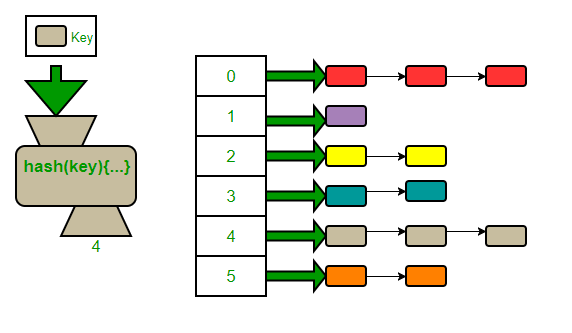
The way I implemented linked lists was with nodes an pointers In a style of a chain where I simply linked one to the other. I used these to store books and used it for my separate chaining.

Red and Black trees:

This was a difficult algorithm to implement and I could not do it as I soon realized as the swapping was too complicated and deletion were quite hard to do. It would take me much more time to complete than I had. So I had to think of an alternative.



Hashing with separate chaining:

In the end this is what I came up with. This method allows me to store books by assigning them addresses which would balance them out through the amount of linked lists that I want. The only issue is that the main difficulty of this algorithm is in the fact that you need to actually come up with a way for the algorithm to provide addresses evenly thus reducing search time. Point would still be fast as I have an address that leads to the column which contains the data I need.

This was more feasible for me as I was able to transform old code and re-purpose it. This was my main goal as hashing with separate chaining. You can tell by the structure that I could re use my nodes and convert them into linked lists that I can store in my ‘buckets’.

Attaining the bucket address:

This is the core difficulty of this project as getting an efficient method going for balancing the addresses evenly in between your bucket is hard. But I got the first part figured out.

What I did was convert the book names into ascii in an attempt to stich it together and modulo it by the amount of buckets.

That did not work as numbers involved were too huge.

So I made a function which broke the ascii collection down into peace’s of modulo which it made into another number that it re-cycled over and over until it was small enough for me to use as an address.

**Testing**

Testing consisted of the following:

* Classical print: This is a timeless function which I used many times. Place it everywhere that you think does not run and it will reveal the truth. It also shows what runs as it is meant to.
* TryCatch: I tried some trying and catching errors at some point but dealing with segmentations faults it did not help.

**Conclusion**

The project was difficult because of small errors here and there which took a lot of time which I over-estimated. Additionally, I wasted half of the time on a algorithm which was hard for me to understand and was very buggy to implement. I had to abandon red and black trees and move on to chaining. I found some success in it but I still lacked:

* Time as the progress was slow.
* Enough knowledge of this algorithm.

**References**

|  |  |
| --- | --- |
| **Source** | **Use** |
| Stackoverflow | Tips on small code. |
| Youtube | Tutorials which I took bits an peace’s from. |
| Kortext | Tutorials, tips and exercises. |
| Google | Used it for quick search ups of what I don’t know. |
|  |  |