CSPB 2270 Term Project Proposal

Justin Lassiter

**Overview:**

My project idea is a solution to a problem that myself and my team at work deal with daily. I work on the finance team at a local bank and one of the daily responsibilities we have as a team is to ensure that certain customers have their deposit accounts collateralized with financial securities owned by the bank. Customer balances change daily, and this requires us to repledge securities daily. This is a very manual process, which requires us to pull a securities listing and customer balance reports. We then must go through each customer balance and check if their balances are adequately collateralized. This sometimes involves unpledging certain securities from a customer and pledging new, larger securities. Additionally, due to regulatory requirements, we need to keep the excess on pledged collateral (security(ies) value – customer account balance(s)) less than a threshold. We typically aim for no more than 50% of the collateralized balances in excess. These changes are then manually added to a tracker to be uploaded/sent to the safekeeper of our securities.

**Project Idea:**

I want to automate this process. My idea on how to do this involves using a Red-Black-Tree. In my mind, the key part of this would be security selection based on the customer’s balance. My thought would be to use a red-black-tree to build a tree based on a securities market value, which is what we use to determine coverage of the account.

The overall high-level steps I’m thinking for this would be the following:

1. Import customer balance data from a daily csv file we receive daily.
   1. Create node structure for each customer holding relevant account detail and store in a map structure.
2. Import the security data from the security detail csv file we receive daily.
   1. I would create a node structure to hold all relevant detail of the security
   2. As it’s being imported, build the red-black tree where it will consistently keep the tree balanced based on market value.
   3. For securities/nodes that are already pledged to a customer when imported, I would put these in their own tree – I want to be able to see/print all pledged securities – maybe just loop through the customers instead? Thinking this is the better route.
   4. As the program is building the tree, for securities already pledged, affiliate these with the customer nodes and remove from the red-black-tree.
3. For all customer nodes that have inadequate collateral, traverse the red-black-tree to search for a security that meets the required capital for the balance. This will be based on criteria/logic laid out below. Once a security is found that meets the criteria, this security will be removed from the tree and placed into the customer’s node. This will cause the red-black-tree to balance itself out as these security assignments progress and are removed from the tree.
   1. **Assignment search and assignment logic**: this is preliminary, but my thought is that each customer node will hold it’s over/under pledged balance. For each of these customer nodes, we can check if the balance is under pledged; if so, unpledged (or add the security node back into the tree) and then perform a search within the red-black-tree to find a security with a value between the needed account value and a 50% margin of the needed value. If it does not find a security within that range, it will loop around with an adjusted, reduced value range at 50% of the needed value up to the needed value. If a security is found in this range, it will be added to the customer node and it will continue to search the tree for additional securities to cover the balance. It will essentially use the needed pledge amount that remains to start the search again. I’m envisioning a series of while loops to perform these searches.
4. Extract all pledge changes as well as current customer details into csv files.

Overall, I think the red-black-tree would be an appropriate data structure to hold the security nodes. This allows for O(logN) time complexity when searching for an appropriate security and will scale nicely, as we have thousands of securities. My initial thoughts on the challenging components of is that: 1. I’ve never implemented a red-black-tree so there will be a slight learning curve to make sure I understand the algorithms / how the tree balances itself. 2. Implementing the logic for security lookup to assignment to a customer will be a challenge. As noted above, I have a general idea of how I could possibly implement it, but I won’t know 100% until I get into it. I’m really looking forward to implementing this for the experience, but it will also make my workday much easier.