Name: Mike Brian Leung

Github: <https://github.com/hkmamike/loanAssigner>

**Affirm Take Home Coding Challenge Write-up**

1. How long did you spend working on the problem? What did you find to be the most difficult part?

It took me about 3 hours to code a working solution, but I will have to spend 1-2 more hours to clean up the code and to do some edge case testing.

Perhaps it was by design, the data sets provided does not cover some covenant situations, so it was challenging to ensure that my code works for those cases.

1. How would you modify your data model or code to account for an eventual introduction of new, as­of­yet unknown types of covenants, beyond just maximum default likelihood and state restrictions?

If the additional covenants involve only simple conditions checking, I only need to add the new properties to the data model, write a new checker function, and introduce the checker function to the processing flow.

If the additional covenants concerns the relationships or states of existing loans already assigned to the candidate facility, we might need to store the sensitive properties of the loan assignments as we process new loans for later reference.

1. How would you architect your solution as a production service wherein new facilities can be introduced at arbitrary points in time. Assume these facilities become available by the finance team emailing your team and describing the addition with a new set of CSVs.

Currently, my code reads input CSVs in the beginning and generates local data models to keep track of facilities. Streaming is simulated by reading through the lines of loans.csv and processing each line.

For a production service, one option is to host these data models on a database and rewrite the loan processing flow as a function that the client can call. We can write a simple parser to update the database as the finance team sends us the CSVs for the new facilities.

1. Your solution most likely simulates the streaming process by directly calling a method in your code to process the loans inside of a for loop. What would a REST API look like for this same service? Stakeholders using the API will need, at a minimum, to be able to request a loan be assigned to a facility, and read the funding status of a loan, as well as query the capacities remaining in facilities.

In a REST API service for this process, instead of iterating to-be processed loans’ data locally, we would have a service that stakeholders can call to request loan processing by sending our server the requested loan’s information. When our server receives requests, it would process them and return the status (and other information) of the requests to the stakeholders. Our server would stand in between the backend data models and the stakeholders’ clients. We would need to have some kind of access control because the funding status of loans and remaining capacities of facilities are not public information. We would also need to ensure that our server does not over subscribe a facility when processing parallel requests, one way to do this is to limit facility assignment access to one processing thread on the server at a time.

1. How might you improve your assignment algorithm if you were permitted to assign loans in batch rather than streaming? We are not looking for code here, but pseudo code or description of a revised algorithm appreciated.

If we can process loans in batches, we can attempt to shuffle-in loans that do not find a matching facility initially. For instance, if we move some loans from facility #13 into #14 or #15, we can probably free up some space in #13 to accommodate loan #425. Pseudo code to illustrate this:

- Identify the facility with the biggest remaining available value (facility X)

- For each of the other facilities that is not completely full (remaining available value > some threshold), find loans in facility X that can fit in the remaining available value and move it over, freeing space in facility X

- Attempt to place unmatched loans into facility X

- Repeat the process for each set of covenant restrictions that apply to outstanding unmatched loans

- Finalize batch processing

1. Because a number of facilities share the same interest rate, it’s possible that there are a number of equally appealing options by the algorithm we recommended you use (assigning to the cheapest facility). How might you tie­break facilities with equal interest rates, with the goal being to maximize the likelihood that future loans streaming in will be assignable to some facility?

If two facilities cost the same, their difference in value to the company comes from the difference in their probability to be utilized. There are two major driving factors that we can consider:

Covenants: facilities that have fewer covenants are more valuable to us as they are less restricted. If we have sufficient data insight, we can even estimate the value (negative) of each covenant base on how likely our typical customer’s loan is going to be restricted by it. For example, a facility that has a covenant banning it from servicing loans in Nevada is probably less valuable than another facility that has a covenant banning it from servicing loans in Cuba. We should assign loans to facilities that are less valuable to us when we can.

Remaining available amount in facility: if we are restricted to service each loan with only one facility, the facilities that have larger remaining available amount are more valuable to us, because they can service loans of bigger sum. For example, we cannot assign a $1,500 loan to a facility that only has $1,000 left in remaining available amount. It is worth mentioning that even if we can service each loan with more than one facility, the value from larger remaining available amount should still be positive because when we split a loan into multiple facilities, we increase our covenant exposure and admin overhead.