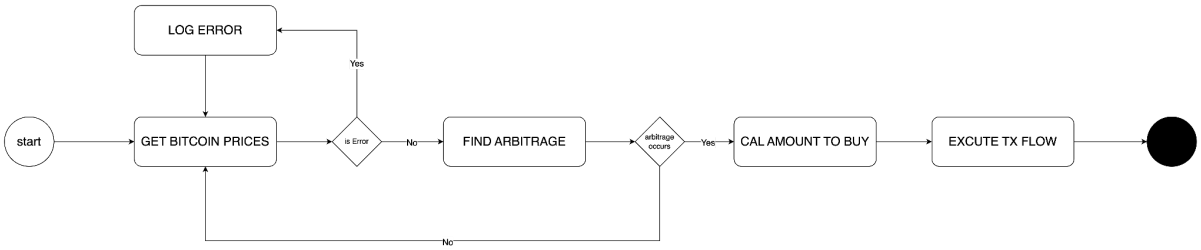
Arbitrage Bot System Design

1. Assumption

The price fetched from CEX is always correct, there's no latency issue. The system has all API keys from CEXs and all accounts in CEXs have enough funds to execute arbitrage.

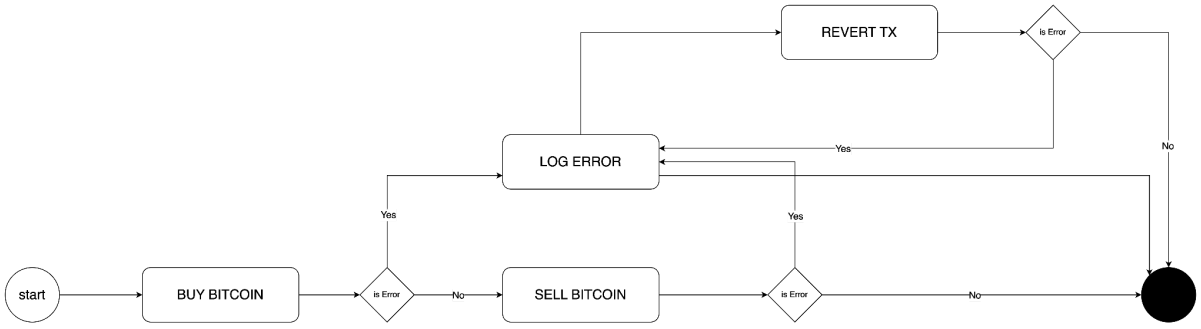
2. Flow Charts

2.a. Find Arbitrage Flow



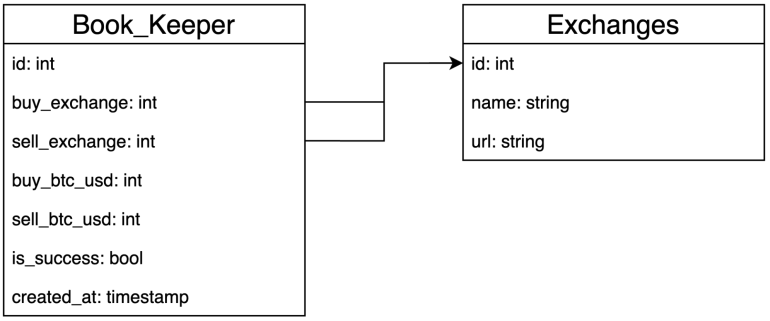
This flow will use a cron job to execute. First of all, the bot is going to fetch bitcoin prices from CEXs, then find if there are arbitrages or not. If so, the bot needs to calculate how many bitcoins should be bought in this round of arbitrage. The execute tx flow will take the following job to finish the arbitrage.

2.b. Execute Tx Flow



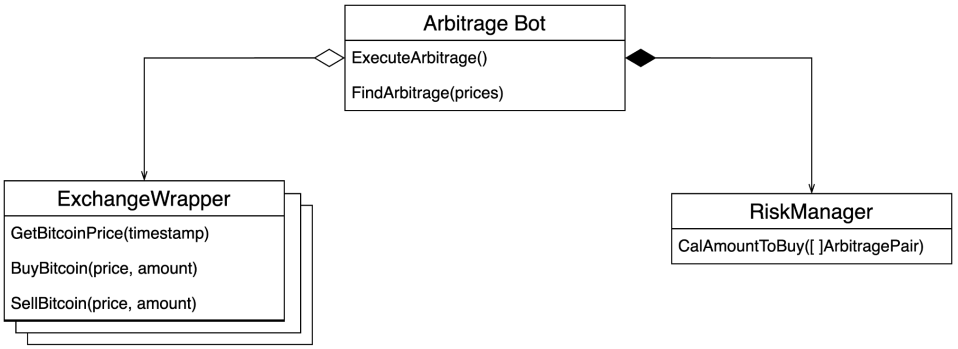
The execute tx flow will buy bitcoin with the given amount and price from the CEX, if error occurs, then abandon this arbitrage, otherwise, processing sell bitcoin to another CEX. If error occurs, then revert the tx by selling the bitcoin to original CEX to stop loss.

3. Database tables



The database is quite simple, I only store a bookkeeper to record all arbitrages no matter it executes successfully or not. This could be displayed in a chart to evaluate the performance of the strategy. The data will be inserted when the bot finds arbitrage and updates the is_success status when the execute tx flow is done.

4. UML Diagram



4.a. ArbitrageBot

The arbitrage bot interface has two functions, execute arbitrage is to execute the program, and find arbitrage will compare all prices from CEXs to look for the arbitrage.

4.b. ExchangeWrapper

The exchange wrapper interface basically wraps CEX apis such as get_bitcoin, buy_bitcoin, sell_bitcoin so that we can call the same function to execute the same behavior.

4.c. RiskManager

The risk manager is the core business logic of the arbitrage bot, cal amount to buy function needs to have a risk factor to calculate if this arbitrage really gains the profit. Because all the way down to the buy and sell bitcoin takes too many times, it definitely causes loss, so the risk manager plays an important role in this system design.

5. Tradeoff

I assume the bot is implemented by Typescript, so there's no multi-threads mechanism, hence, it would take much time to get all prices from CEXs. To compensate for this time loss, I don't spare the execute tx part to Message Queue, otherwise, it would take much more time to execute the tx. If the bot can be implemented by another programming language which supports multi-threads mechanism, it will shorten the fetching prices time by using thread pool.