DSA Project Report

VII. Currency

Team 49

Data Structures Used:

- Hash Tables
- Singly linked lists
- Arrays
- Priority Queue (Min Heap)
- Graph (Using Adjacency list)

Functions Used and Their Time Complexities:
(V is the number of currencies in a trade bank)
(E is the number of conversions in a trade bank)
(e is the number of conversions in the adjacency list of a currency)
(T is the number of trade banks)
(P is the number of edges in a path)
(C is the number of currencies in a cycle)

- Add a trade bank – Using hash tables for accessing ~ O(1)

- Delete a trade bank - Using hash tables for accessing ~ O(1) And then freeing all the allocated memory in the trade bank (currencies and the adjacency lists) ~ O(V + E)

- Add a currency – Using hash tables for accessing trade bank as well as currency ~ O(1)

- Delete a currency - Using hash tables for accessing trade bank as well as currency ~ O(1) and then

    - freeing the allocated memory to the adjacency list as well as the currency node itself
    - freeing the memory allocated to this currency node in the adjacency lists of all the other currencies of the trade bank

  (Both steps cumulatively take ~ O(V + E)

- Add a conversion – Using hash tables to access the trade bank and both the src and dest currencies ~ O(1) and then traversing the adjacency list of the src currency to check if a conversion already exists ~ O(e)
  (and insertion at front if the conversion does not exist which takes O(1))

- Delete a conversion - Using hash tables to access the trade bank and both the src and dest currencies ~ O(1) and then traversing the adjacency list of the src currency to check if the conversion exists ~ O(e)
  (If the conversion exists, then delete it, which takes O(1))

- Find the Best Path –
    - We use Dijkstra's Algorithm which is optimized using priority queues and find the best path in every individual trade bank and then compare those paths obtained to get the final best path from the specified src currency to the dest currency
    - To find the best path in a single trade bank, it takes ~ O(V logV + E logV) --> ~ O(E log V)
    - We also print the best path which takes ~ O(P) (this time is negligible to the time taken for finding the path)
    - Thus, since we are finding the best path for T trade banks, the overall complexity is ~ O (TE log V)

- Find the second Best Path –

    - For each trade bank - We first find the best path between src and dest. Then we start deleting the edges in the best path one at a time, and calculate the best path for the updated graph each time, and update the 2nd best path.
    - Calculating the best path in a trade bank takes ~ O(E logV) and since we are doing this for P times, hence finding the second best path in a trade bank takes ~ O(PE log V).
    - Since we are doing this for T trade banks, the overall complexity of finding the 2nd best path from src currency to the dest currency takes ~ O(TPE log V)

- Finding a cycle in a trade bank –
    - We perform DFS in conjunction with Union-Find method for finding cycles
    - We start with performing DFS on each currency node whilst updating the "king" values of the various nodes. This DFS visit takes ~ O(V + E)
    - And once we have found a cycle, we also print it, which takes ~ O(C) (this time is negligible as compared to the time for finding a cycle)
    - Hence the overall complexity for finding a cycle in a trade bank is ~ O(V + E)

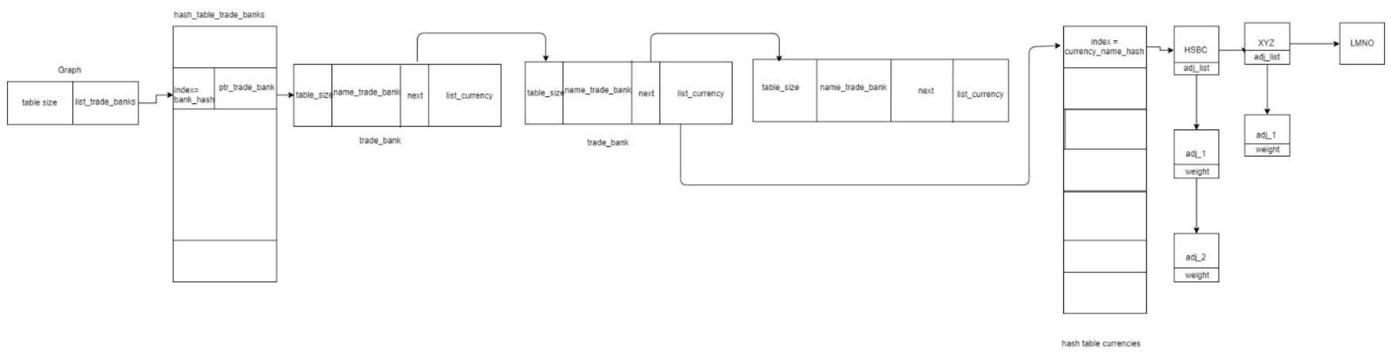
Algorithms Used:

- Dijkstra's Algorithm (Using priority queues)
- DFS
- Union-Find

<u>Division of Work</u>:
(The currency.h file was discussed and created as a joint effort in through the means of Microsoft teams)

- Harish (2020102067) – add_currency, delete_currency, input_generator.
- Sudheer (2020101065) – delete_trade_bank, print_currency, best_path, find_cycle
- Ruhul (2020102031) – add_trade_bank, print_trade_bank, find_cycle, DFS
- Tanmay (2020112017) – add_conversion, delete_conversion, DFS, user_main
- Vithesh(2020115002) – Priority_queue, best_path, second_best_path, user_main

<u>Link to Mind Map of the ADT</u> (Diagrammatic representation):



Click here to view the bigger (interactive) version of the diagram.

<u>Explanation</u>:

- The ADT consists of 2 types hash tables, one to store the trade banks and the other one to store the currencies in each trade bank.
- In the graph struct, the list_trade_banks points to the hash table containing the trade banks.

- Each of the trade banks contains a next pointer which points to the next trade bank stored at the same hash value in case of a collision. (separate chaining)
- Similarly, the list_currency in each trade bank points to the hash table containing the currencies.
- Each currency node contains 2 pointers
    - A next pointer which points to the next currency node stored at the same hash value in case of collisions (separate chaining)
    - An adj_list pointer which points to a linked list which is the adjacency list of the currency node
    - Each node in the adjacency list stores the conversion rate of the conversion from the source currency to that respective currency specified by that node.