COL761 A1 - Transactional Data Compression

2020CS50423	2020CS50440	2020EE30313
Dharmeswar Basumatary	Sahil Shahare	Vibhor Sengar

Data Compression Algorithm:

Input:

- **Transactions**: List of transactions (lists of items).
- **Support threshold**: Minimum support count for an item set to be considered frequent.

Generate Frequent Item Sets using FP-Tree:
(Reference with code: https://github.com/integeruser/FP-growth)

- Build the initial header table containing the frequency of each unique item in the transactions.
- Sort items in the header table based on their frequency in decreasing order.
- Build the **FP-Tree:**
    1. For each transaction, inserted items in the transaction into the tree, following the order of the sorted header table.
    2. Created conditional FP-Trees for each item, starting from the least frequent item.
- Mined the FP-Tree for frequent item sets by recursively traversing the tree from the least frequent item to the most frequent item. Store these frequent item sets.

Encrypt the Original Transactions:

- For each frequent item set assigned a unique ID and grouped frequent item sets by their length.
- For each transaction:
    1. Starting from the largest frequent item set, check if the current frequent item set is a subset of the transaction.
    2. If the frequent item set is a subset of the transaction:
        - Put the unique key for this frequent item set in the transaction.
        - Recurse on the (transaction) / (frequent item set).
- Discard the frequent item sets which did not find to be a subset.

Output:

- **Compressed transactions**: List of transactions with item sets replaced by encryption keys.
- **Encryption map**: Mapping of unique keys to corresponding frequent item sets.

Data Decompression Algorithm:

Input:

- **Compressed transactions**: List of transactions with encryption keys
- **Encryption map**: Mapping of unique keys to corresponding frequent item sets.

Decrypt the Transactions:

- For each transaction in encrypted transactions, replace unique keys with their corresponding frequent item sets using the encryption map.

Output:

- **Original transactions**: List of transactions with frequent item sets restored

Important notes:

- Ensuring that the encryption keys generated for each frequent item set are sufficiently strong and unique.
- The order of generating encryption keys are deterministic for both compression and decompression.
- Making sure to handle scenarios where the same item set might have been encrypted using different keys due to variations in the order of processing.