FP-Growth Algorithm

Key Points

- **Definition**: FP-Growth is an algorithm used for mining frequent itemsets in transactional databases. It is an alternative to the Apriori algorithm and is known for its efficiency and scalability.
- **Purpose**: To discover frequently occurring patterns or itemsets in large datasets, which can be useful for association rule learning and market basket analysis.
- **Advantages**:
  - **Efficiency**: FP-Growth can handle large datasets more efficiently than Apriori by avoiding candidate generation and multiple database scans.
  - **Compact Representation**: Uses a compact data structure called the FP-tree to represent the dataset, which reduces memory usage and computational complexity.
- **Data Structure**:
  - **FP-tree (Frequent Pattern Tree)**: A tree structure that stores compressed information about frequent itemsets. Each node represents an item, and the tree branches represent itemset combinations.

Algorithm

1. **Create the FP-tree**:
   - **Scan Dataset**:
     - Scan the entire dataset to count item frequencies and identify frequent items based on a minimum support threshold.
   - **Build the FP-tree**:
     - Create a root node and insert items into the FP-tree, maintaining item frequency counts and conditional paths. Each path in the tree represents a transaction.
2. **Mine the FP-tree for Frequent Patterns**:
   - **Conditional Pattern Base**:
     - For each item in the FP-tree, construct its conditional pattern base, which is a sub-database containing all transactions that include the item.
   - **Construct Conditional FP-tree**:
     - Build a conditional FP-tree for each item based on its conditional pattern base. This tree represents itemsets conditional on the presence of the item.
   - **Extract Frequent Patterns**:
     - Recursively mine the conditional FP-trees to extract frequent itemsets. Each conditional FP-tree yields frequent itemsets for the item.
3. **Combine Results**:
   - Combine the frequent itemsets discovered from all conditional FP-trees to get the complete set of frequent itemsets in the dataset.
4. **Generate Association Rules** (if needed):
   - Based on the frequent itemsets, generate association rules that reveal relationships between items (e.g., "If a customer buys bread and butter, they are likely to buy milk").