**Theoretical description of the algorithm:**

1. support：It is the ratio of the number of simultaneous occurrences of A and B to the total number of transactions in all transactions. To put it another way, in the real data, the proportion of the association between A and B is supported. Calculated using the following formula:C:\Users\dhy\AppData\Roaming\Tencent\Users\3480681912\TIM\WinTemp\RichOle\KE[WXI6(9(W4G%~Z]}[)X)S.png

Where P(A∪B) means the ratio of A and B in the transaction;

1. confidence：The confidence of A⇒BA⇒B refers to the proportion of firms that contain A and also contains B. Calculated using the following formula:

confidence(A⇒B)=P(B|A)

For example, a total of 100 transactions, 50 contain A, and 20 of these 50 transactions also contain B, then, confidence (A ⇒ B) = 40%

1. Strong rules: We define association rules between entities as strong rules, if the relative support and confidence between entities meet our predefined minimum support threshold (min\_sup) and minimum confidence. Degree threshold (min\_conf).
2. Frequent itemsets: Refers to the set of items that frequently appear in a transaction. The criterion for "frequent" is that the number of occurrences of this item set satisfies the minimum support count (threshold).

You can get a conclusion: the strength of the association rules between entities can be determined by their relative support and confidence, and these two indicators can pass the absolute support: support (A) support (A) and support (A ∪ B )support(A∪B) to calculate. Therefore, association rule mining can be transformed into frequent item set mining.

Thus, the mining of association rules is a two-step process:

1. Find all frequent itemsets: According to the relative support degree, the definition of confidence can be known that if there are strong association rules between any two entities, they must exist in the frequent itemsets. Otherwise, if these two entities Does not exist in frequent itemsets, it will not produce strong association rules

2. Generate strong association rules from frequent itemsets: calculate support and confidence, find strong rules between entities

Obviously, when we determine the entity to analyze, the cost of the second step is very small. The key is the first step: mining frequent itemsets. The Apriori algorithm solves this problem.

The translation of Apriori into Chinese is “a priori”, so the a priori nature is the core of the entire Apriori.

Theorem 1: a priori nature: all non-empty subsets of frequent itemsets must also be frequent.

Explanation: For example, if an item set {I1, I2, I3}{I1, I2, I3} is frequent, then the number of simultaneous occurrences of these three items is greater than the minimum support count, so we can infer that it The support count of any non-empty subset, {I1}{I1}, {I2, I3}{I2, I3}, etc. must also be larger than the predefined threshold, so it is frequent.

Conversely, if an item set II is frequent, then adding an item AA to this item set, the new item set {I∪A}{I∪A} will at least not be more frequent than II because Something, so the number of simultaneous occurrences of all items in the item set will not increase.

Think further: If item set II is infrequent, then no matter what item or number of items are added to it, he will not become a frequent item set. This special nature is also called "anti-monotonicity." We will replace this "anti-monotonicity" with the following theorem 2:

Theorem 2: Anti-monotonicity: An item set, if at least one non-empty subset is infrequent, then this item set must be infrequent.

It is the use of the above theorem 1, theorem 2, Apriori is designed, it generates a frequent kk item set from the frequent k−1k−1 item set through the layer-by-layer search mode, and finally gets all the frequent itemsets.

It can be seen that the core component of Apriori is how to generate frequent kk itemsets from frequent k−1k−1 itemsets. Why do you want to find frequent kk items in this way? Through Theorem 1, 2, we know that if an item set is a frequent kk item set, then its arbitrary non-empty subset must be frequent, so if we find all the frequent k−1k−1 itemsets now, Then the frequent kk item set must be generated by the combination of these frequent k−1k−1 items. Can be reduced to the following two theorems:

Theorem 3: Any frequent kk item set is generated by a combination of frequent k−1k−1 items.

Theorem 4: All k−1 subsets of frequent k itemsets must be all frequent k−1 itemsets

Key code for implementation:

parameter:

1. D: Transaction database, expressed in the form of a dictionary. The key is a TID, which is a collection of items that appear in the transaction, and this collection is stored as a list. Form like {TID: [I1, I2]}

2. min\_sup: minimum support count threshold, int type, in the code, set to 2

Simple function:

1. find\_frequent\_1\_itemsets(D, min\_sup): traverse D, find all frequent 1 item sets according to min\_sup, the result is recorded as L1, L1 is list type, and each element is a list of length 1. Use steps 1 and 2 above to find out the frequent 1 item set.

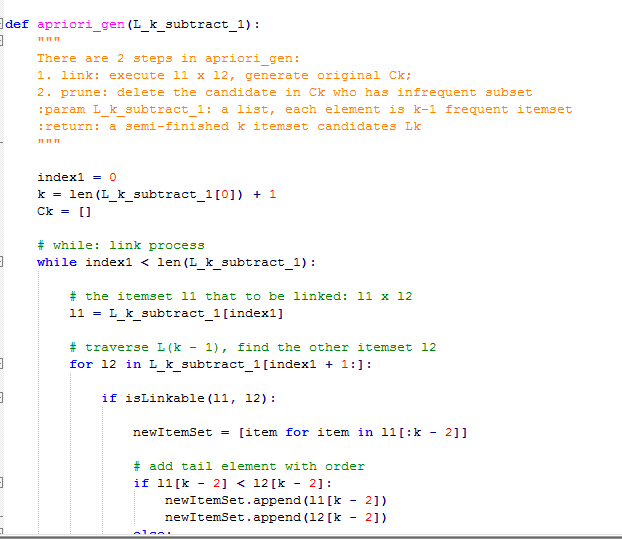
2. isLinkable(l1, l2): l1, l2 are list type. The function of this function is to judge whether two sorted items are set l1, l2 is connectable, used in step 3 above, the premise of connection .

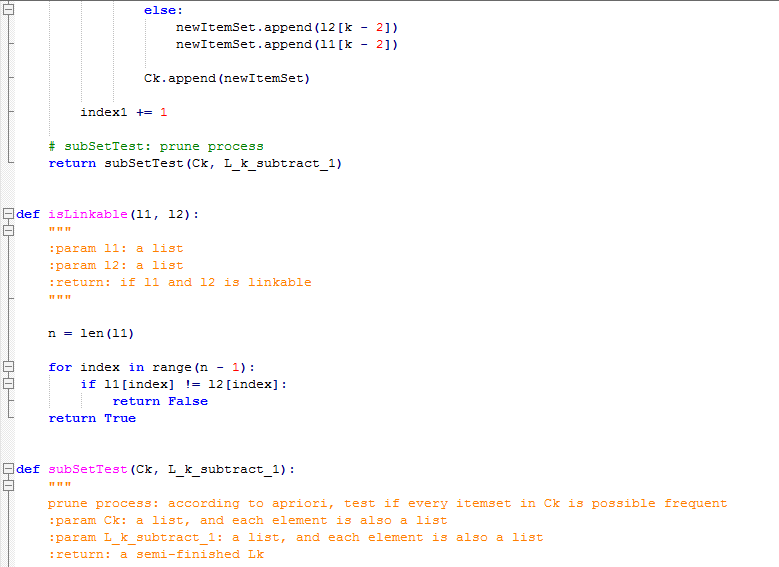
3. gen\_ksub1\_subsets(s): Generates a subset of the set s of length k - 1, s is a list type, the generated result is a list type, and each element of the result is a list type. This function is used in step 4 above - pruning to determine if all k - 1 subsets of a k-item set are frequent

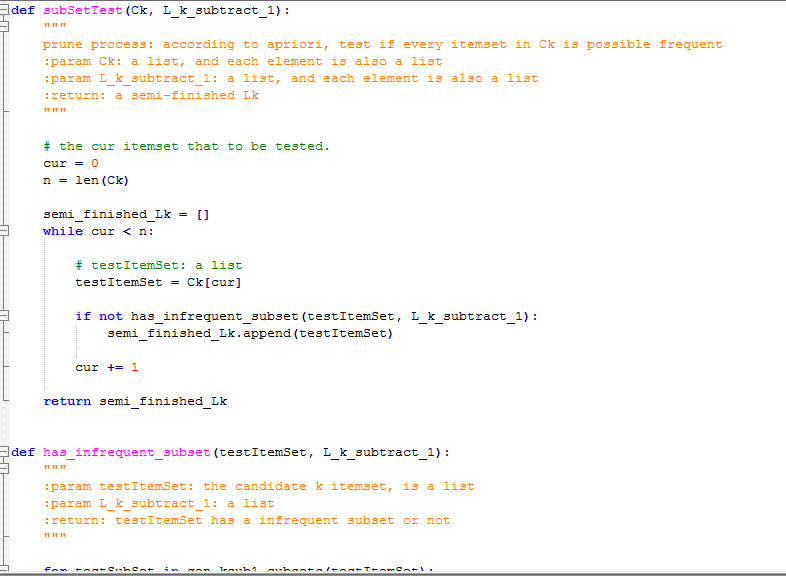
4.subsets(S): find all subsets of the set S, S is a list type, the result is a list, and each element of the result is also a list type, representing a subset of S. This function is used in the filtering of step 5 above to find a subset of each transaction.

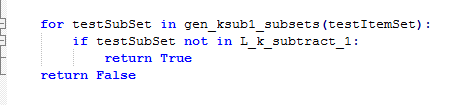
The implementation code of the key function:

1. The first is two steps of connecting and pruning:

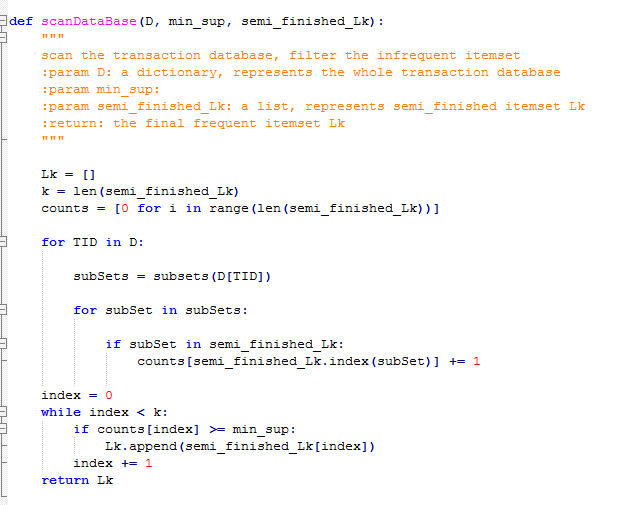
The function that implements these two steps is apriori\_gen(L\_k\_subtract\_1), and the parameter L\_k\_subtract\_1 represents the set of frequent k−1k−1 itemsets.



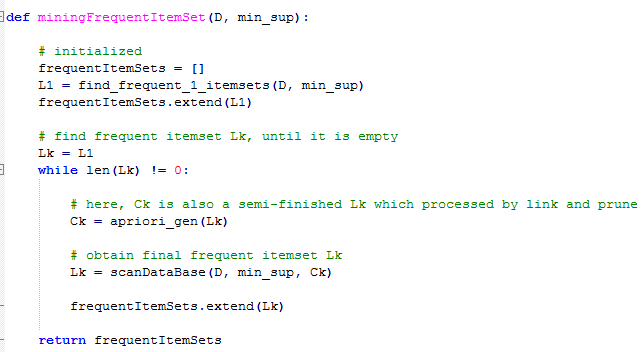




Apriori\_gen also calls the function subSetTest. The function of this function is to do the subset test, which is the process of pruning. For the candidate set Ck, pruning is filtered according to L\_k\_subtract\_1. SubSetTest calls the function has\_infrequent\_subset to check the frequency of each subset. Other functions used, such as isLinkable(), gen\_ksub1\_subsets() are omitted.

Finally, one more step is to scan the entire database, filter the candidate set twice, and find the frequent item set LkLk. The code is as follows:

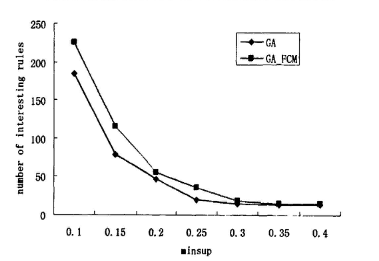
The function subsets() used is omitted, and the function of the subset is also very classic and easy to find.

Based on these, the main function can be given:

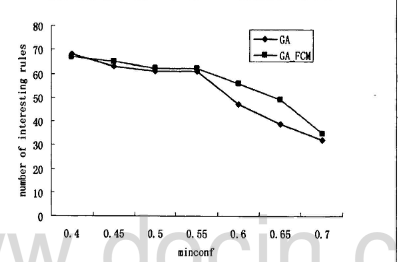
By finding out all the frequent itemsets, we can calculate the entity with strong association rules (the entity may be an item or an item set) through the formula of support and confidence that we said above.

**Test description:**

1. 给定置信度为80%，关联规则数目随支持度变化的曲线图



2）给定支持度为30%，关联规则数目随置信度变化的曲线图



3）给定置信度为80%，请确定某个支持度s使得获取的关联规则数目正好大于20，请输出s和所获取的关联规则数目

S=0.08

Number of association rules =25