CS 145 - homework 1

Zooey Nguyen zooeyn@ucla.edu April 18, 2021

Question 1.

(a) Step by step iterations of the apriori algorithm, three steps per level. First construct the self-joined candidate set $L_{k-1}*L_{k-1}$, then prune to get candidate set C_k , then scan database and find L_k . Repeat until C_k or L_k is empty. Level 1 begins with the exception that the candidate set is constructed with no pruning by scanning database for all items.

```
C_{1} = \{a, b, c, d, e, f, g, h, i, j, k, o\}
L_{1} = \{a, b, c, d, e, f, h, j\}
L_{1} \times L_{1} = \{ab, ac, ad, ae, af, ah, aj, bc, bd, be, bf, bh, bj, cd, ce, cf, ch, cj, de, df, dh, dj, ef, eh, ej, fh, fj, hj\}
C_{2} = \{ab, ac, ad, ae, af, ah, aj, bc, bd, be, bf, bh, bj, cd, ce, cf, ch, cj, de, df, dh, dj, ef, eh, ej, fh, fj, hj\}
L_{2} = \{aj, bc, bd, bh, bj, cj, hj\}
L_{2} \times L_{2} = \{abc, abd, abh, abj, acd, ach, acj, adh, adj, ahj, bcd, bch, bcj, bdh, bdj, bhj, cdh, cdj, chj, dhj\}
C_{3} = \{bcj, bhj\}
L_{3} \times \{bcj, bhj\}
L_{3} \times \{bcj, bhj\}
L_{4} = \{bchj\}
C_{4} = \emptyset
\cup_{k} L_{k} = \{a, b, c, d, e, f, h, j, aj, bc, bd, bh, bj, cj, hj, bcj, bhj\}
```

- (b) The database got scanned three times, that is, as many times as it took to construct an L_k set from its candidate set C_k .
- (c) The maximal itemsets are e, f, aj, bd, bcj, bhj.

 The closed itemsets are b, c, d, e, f, j, aj, bc, bd, bh, bj, cj, bcj, bhj.
- (d) Step by step iteration of the apriori algorithm, this time when we look for frequent itemsets we need to also note which ones have a max price of less than 40 and eliminate them after using them in the

self-join. The ones to eliminate will be marked with asterisks.

$$C_{1} = \{a, b, c, d, e, f, g, h, i, j, k, o\}$$

$$L_{1}* = \{a*, b*, c, d*, e*, f*, h, j*\}$$

$$L_{1} = \{c, h\}$$

$$L_{1} \times L_{1} = \{ab, ac, ad, ae, af, ah, aj, bc, bd, be, bf, bh, bj, cd, ce, cf, ch, cj, de, df, dh, dj, ef, eh, ej, fh, fj, hj\}$$

$$C_{2} = \{ab, ac, ad, ae, af, ah, aj, bc, bd, be, bf, bh, bj, cd, ce, cf, ch, cj, de, df, dh, dj, ef, eh, ej, fh, fj, hj\}$$

$$L_{2}* = \{aj*, bc, bd*, bh, bj*, cj, hj\}$$

$$L_{2} = \{bc, bh, cj, hj\}$$

$$L_{2} \times L_{2} = \{abc, abd, abh, abj, acd, ach, acj, adh, adj, ahj, bcd, bch, bcj, bdh, bdj, bhj, cdh, cdj, chj, dhj\}$$

$$C_{3} = \{bcj, bhj\}$$

$$L_{3}* = \{bcj, bhj\}$$

$$L_{3}* = \{bcj, bhj\}$$

$$L_{3} = \{bchj\}$$

$$C_{4} = \emptyset$$

$$\cup_{k} L_{k} = \{c, h, bc, bh, cj, hj, bcj, bhj\}$$

Question 2.

Please note that my tree representation is text-based. Children nodes are listed one indent further than their immediate parents, underneath their parents. For example, the children of the root node are indented once. I also list counts right next to the respective items.

(a) First construct the fp-list:

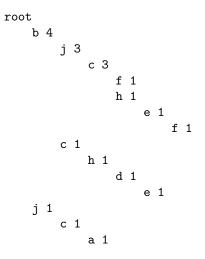
Item	Support
b	7
j	6
c	5
a	4
h	4
d	3
e	3
f	3

Then construct the fp-tree:

```
root
    b 7
         d 1
         j 5
             c 3
                  f 1
                  h 1
                       e 1
                           f 1
             a 2
                  h 2
                       d 1
         c 1
             h 1
                  d 1
    j 1
             a 1
    a 1
         e 1
             f 1
```

(b) The database is scanned two times, one to construct the fp-list and another to build the tree transaction-by-transaction.

(c) Get the conditional database with just the branches that have c. The tree prunes bd, the bjahd branch, and aef. The remaining tree has



In this tree the frequent itemsets with c without a, d, e, f, h include $\{b, j, c, bj, jc, bc, bjc\}$

Question 3.

(a) There are $\boxed{5}$ elements in the sequence, but the length of the sequence is $\boxed{7}$ because some events have multiple items. The number of nonempty subsequences depends on the length of the sequence and is $C(7,1) + C(7,2) + C(7,3) + C(7,4) + C(7,5) + C(7,6) + C(7,7) = 2^7 - 1$ in total, which is $\boxed{127}$.

- (b) We get the self-join set $\{(ab)c, (ab)b, (ab)d, abc, abd, bcd\}$. However we need to prune (ab)b since bb infrequent, (ab)d since ad infrequent, abd for the same reason, and bcd since cd infrequent. So we get $\boxed{\{(ab)c, abc\}}$.
- (c) We get the suffix database:

(d) The length-2 sequential patterns with prefix b are identified by the frequent patterns in the suffix database. The elements a, c, d, and f are frequent here, so we have frequent 2-patterns $bar{ba}{ba}$.