MSC-BDT5002, Fall 2018
Knowledge Discovery and Data Mining

Assignment 1

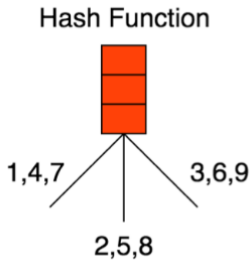
Deadline: Sep. 28th, 11:59pm, 2018

Name:	**ZHENG Dongjia**
ITSC:	**dzhengah**
SID:	20546139
Email:	dzhengah@connect.ust.hk
Director:	Prof.CHEN Lei

Q1. Hash Tree (40 marks)·

Suppose we have 35 candidate item sets of length 3:
{1 2 4}, {1 2 9}, {1 3 5}, {1 3 9}, {1 4 7}, {1 5 8}, {1 6 7}, {1 7 9}, {1 8 9} {2 3 5}, {2 4 7}, {2 5 6}, {2 5 7}, {2 5 8}, {2 6 7}, {2 6 8}, {2 6 9}, {2 7 8} {3 4 5}, {3 4 7}, {3 5 7}, {3 5 8}, {3 6 8}, {3 7 9}, {3 8 9}, {4 5 7}, {4 5 8}, {4 6 7}, {4 6 9}, {4 7 8}, {5 6 7}, {5 7 9}, {5 8 9}, {6 7 8}, {6 7 9}

The hash function is shown in the figure below.



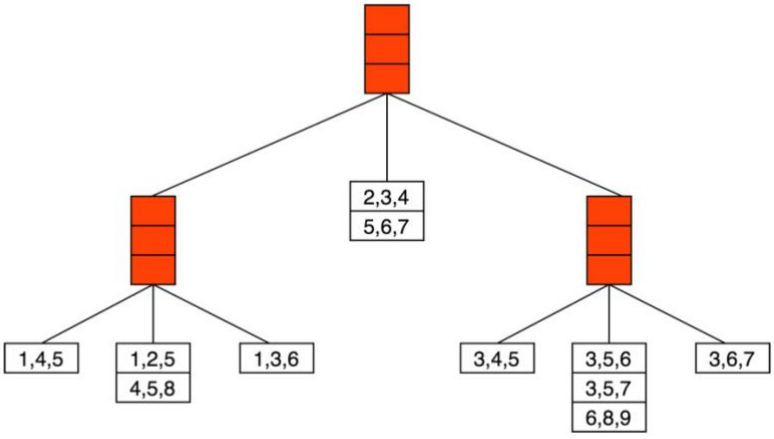
(a) Please write a program to generate a hash tree with **max leaf size 3**, output the nested list (or nested dict) of the hash tree **hierarchically** and draw the structure of the hash tree (you can write program to draw this hash tree or just manually draw it according to the nested list you output). Please write the nested list/dict and the hash tree together in the **A1_itsc_stuid_answer.pdf.**
(35 marks)

- Give an example:

The nested list is (underline is just to make the structure clearer, you don't need to draw it in your assignment):

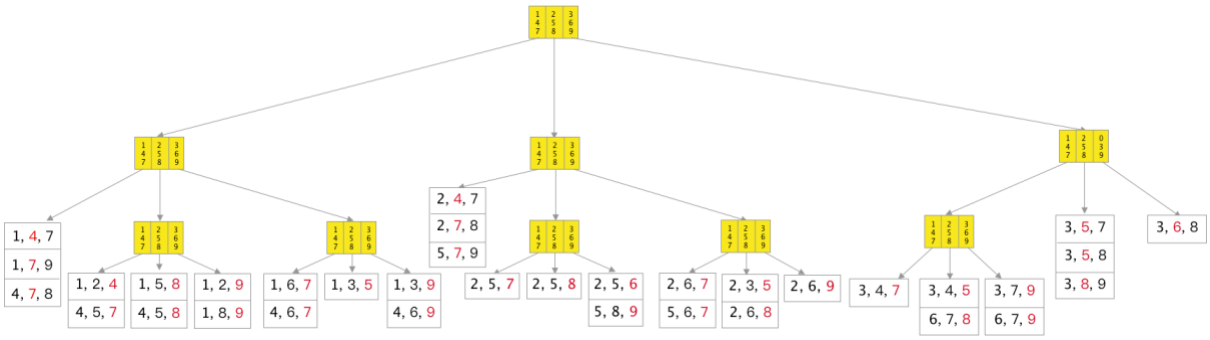
[[[1,4,5], [[1,2,5],[4,5,8]], [1,3,6]], [[2,3,4], [5,6,7]], [[3,4,5], [[3,5,6], [3,5,7], [6,8,9]], [3,6,7]]]

The corresponding hash tree is:



Answer:

The overview of the hash tree:



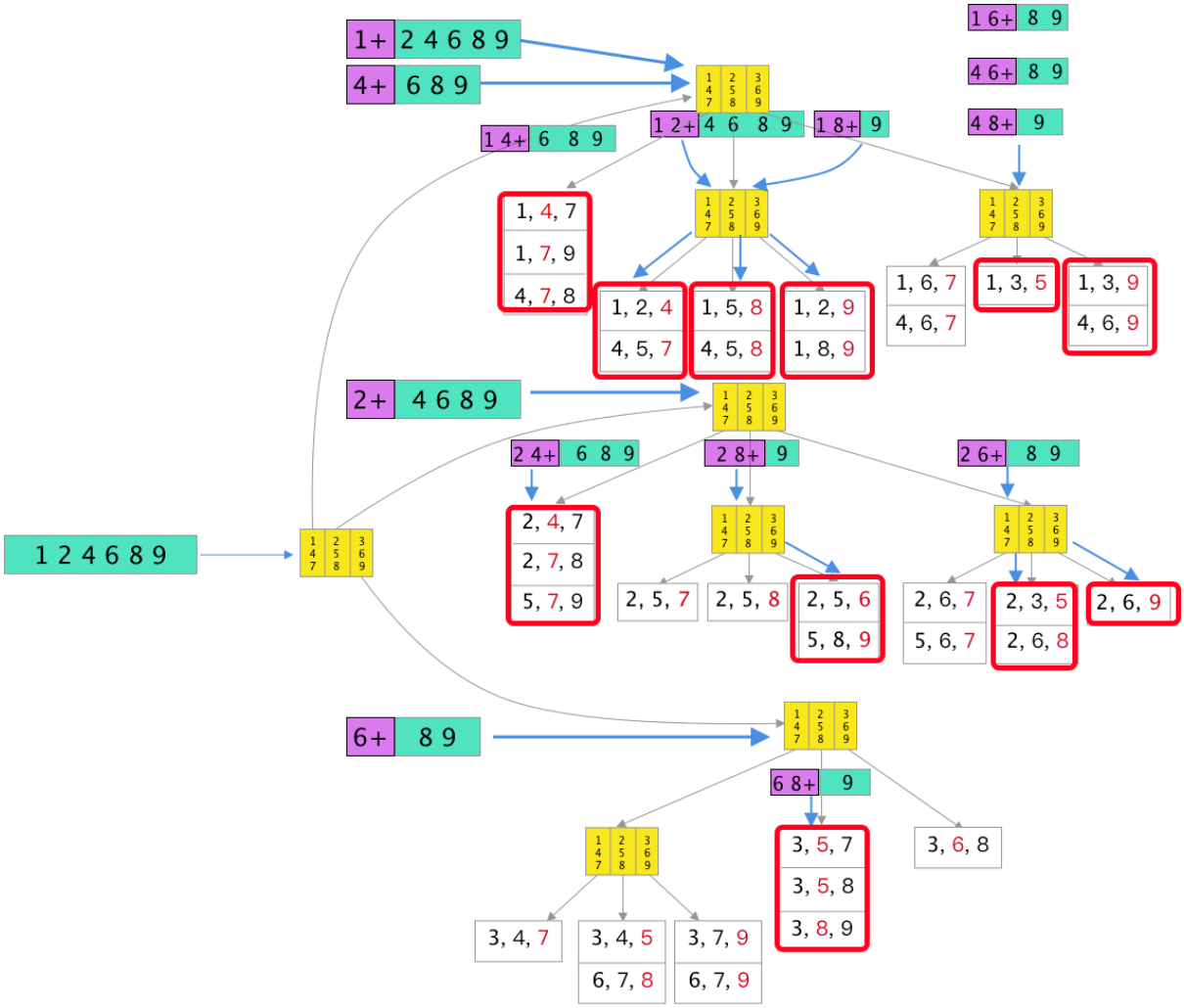
Nested list(in a hierarchical structure):

```
[
    [
        [[1, 4, 7], [1, 7, 9], [4, 7, 8]],
        [
            [[1, 2, 4], [4, 5, 7]],
            [[1, 5, 8], [4, 5, 8]],
            [[1, 2, 9], [1, 8, 9]]
        ],
        [
            [[1, 6, 7], [4, 6, 7]],
            [1, 3, 5],
            [[1, 3, 9], [4, 6, 9]]
        ]
    ],
    [
        [[2, 4, 7], [2, 7, 8], [5, 7, 9]],
        [
            [2, 5, 7],
            [2, 5, 8],
            [[2, 5, 6], [5, 8, 9]]
        ],
        [
            [[2, 6, 7], [5, 6, 7]],
            [[2, 3, 5], [2, 6, 8]],
            [2, 6, 9]
        ]
    ],
    [
        [
            [3, 4, 7],
            [[3, 4, 5], [6, 7, 8]],
            [[3, 7, 9], [6, 7, 9]]
        ],
        [[3, 5, 7], [3, 5, 8], [3, 8, 9]],
        [3, 6, 8]
    ]
]
```

(b) Given a transaction that contains items {1, 2, 4, 6, 8, 9}, how many comparisons are needed using the hash tree which you generate above? Please circle these candidates in the hash tree. No programming required. (5marks)

Answer:

23 comparisons are needed using the hash tree.



Match transaction against 23 out of **35** candidates

Q2. FP-Tree (60 marks)

Frequent Pattern Mining is very important for the retail industry to increase profits. Suppose you are the owner of a grocery, there is a sale records of your store.

Data Description:

groceries.csv: This is a .csv file that contains totally 9835 records and each record records every single transaction in the grocery store. The following table is an example of it.

1	beef
2	butter, sugar, fruit/vegetable juice, newspapers
3	frankfurter, rolls/buns, soda
4	packaged fruit/vegetables
...	...

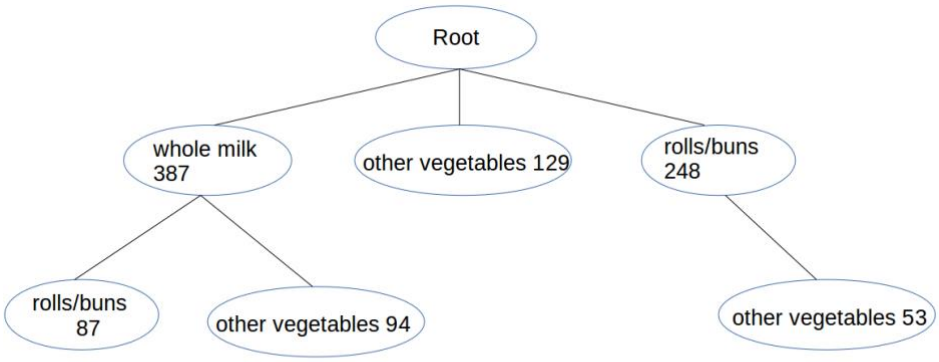
sample_submission.csv: This is a sample submission format for Question (a), you should follow this template to save your result.

Questions:

(a) Please write a program to implement FP-growth algorithm and find all frequent itemsets with support >= 300 in the given dataset. (42 marks)

(b) Based on the result of (a), please print out those FP-conditional trees whose height is larger than 1. (18 marks)

l Give an example of problem (b)'s output: For the tree as follows:



We expect you print the result like :

["Null Set 1", [["whole milk 387",["rolls/buns 87","other vegetables 94"]], "other vegetables 129", ["rolls/buns 248",["other vegetables 53"]]]

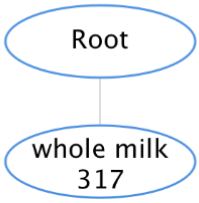
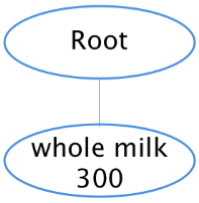
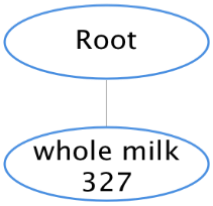
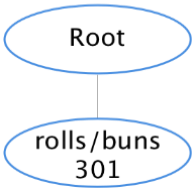
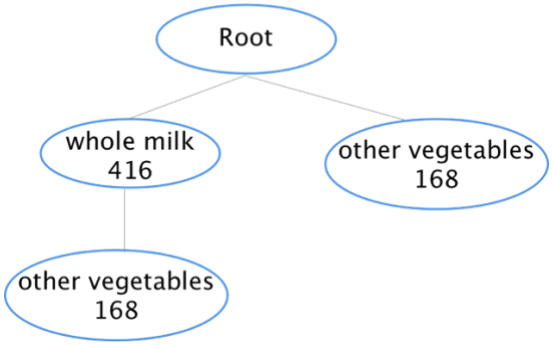
Answers:

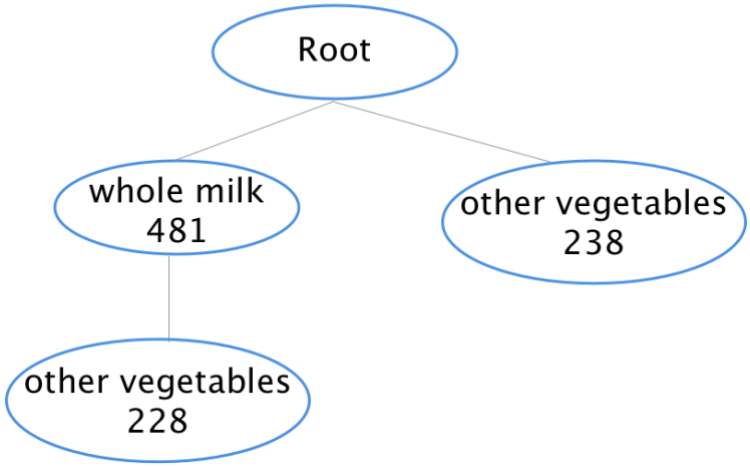
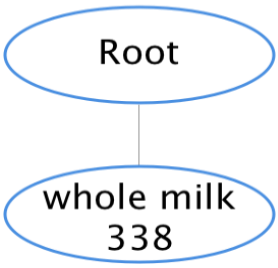
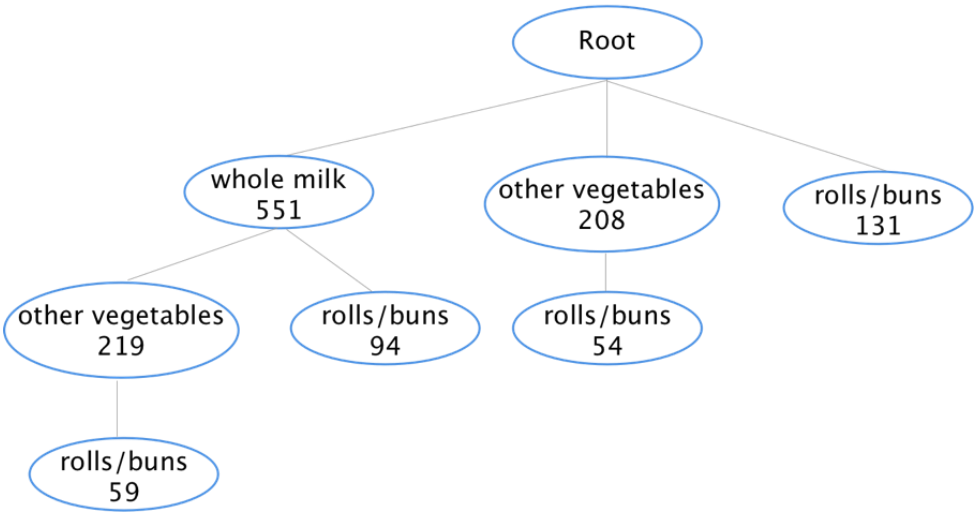
(a) The frequent item sets with support >= 300 in the given dataset.

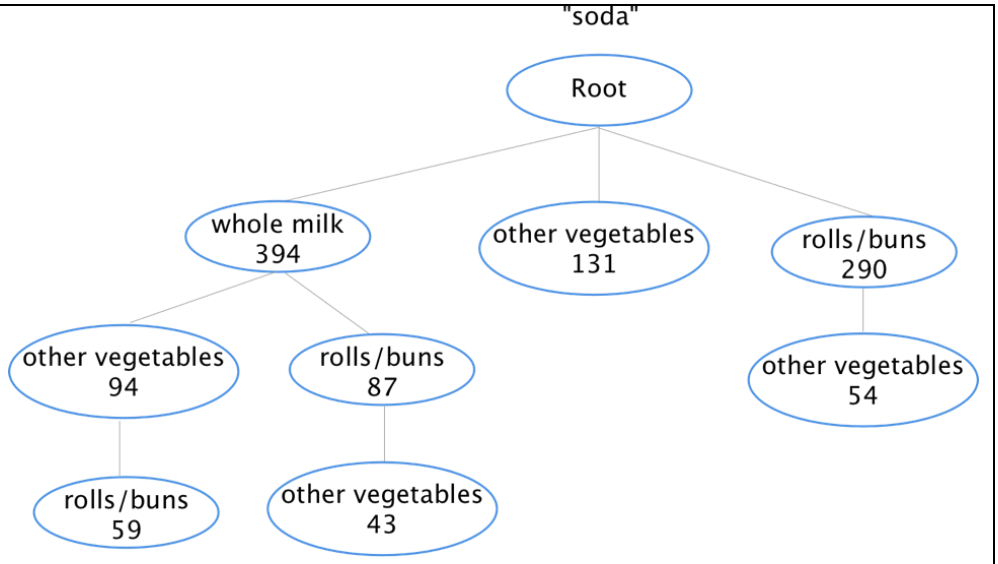
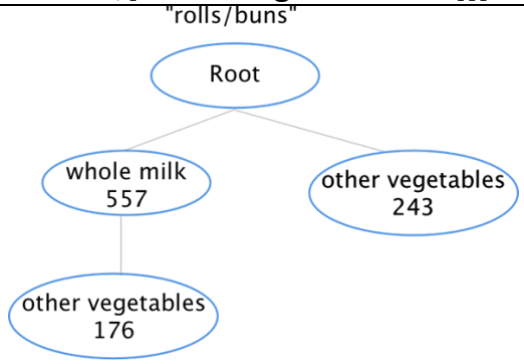
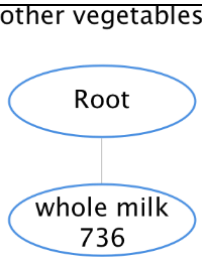
1.	{'onions'}	2.	{'bottled beer'}
3.	{'hygiene articles'}	4.	{'citrus fruit'}
5.	{'hamburger meat'}	6.	{'citrus fruit', 'whole milk'}
7.	{'berries'}	8.	{'pastry'}
9.	{'UHT-milk'}	10.	{'whole milk', 'pastry'}
11.	{'sugar'}	12.	{'sausage'}
13.	{'dessert'}	14.	{'sausage', 'rolls/buns'}
15.	{'long life bakery product'}	16.	{'shopping bags'}
17.	{'salty snack'}	18.	{'tropical fruit'}
19.	{'waffles'}	20.	{'tropical fruit', 'other vegetables'}
21.	{'cream cheese '}	22.	{'whole milk', 'tropical fruit'}
23.	{'white bread'}	24.	{'root vegetables'}
25.	{'chicken'}	26.	{'other vegetables', 'root vegetables'}
27.	{'frozen vegetables'}	28.	{'whole milk', 'root vegetables'}
29.	{'chocolate'}	30.	{'bottled water'}
31.	{'napkins'}	32.	{'whole milk', 'bottled water'}
33.	{'beef'}	34.	{'yogurt'}
35.	{'curd'}	36.	{'yogurt', 'rolls/buns'}
37.	{'butter'}	38.	{'yogurt', 'other vegetables'}
39.	{'pork'}	40.	{'whole milk', 'yogurt'}
41.	{'coffee'}	42.	{'soda'}
43.	{'margarine'}	44.	{'other vegetables', 'soda'}
45.	{'frankfurter'}	46.	{'rolls/buns', 'soda'}
47.	{'domestic eggs'}	48.	{'whole milk', 'soda'}
49.	{'brown bread'}	50.	{'rolls/buns'}
51.	{'whipped/sour cream'}	52.	{'other vegetables', 'rolls/buns'}
53.	{'whole milk', 'whipped/sour cream'}	54.	{'whole milk', 'rolls/buns'}
55.	{'fruit/vegetable juice'}	56.	{'other vegetables'}
57.	{'pip fruit'}	58.	{'whole milk', 'other vegetables'}
59.	{'canned beer'}	60.	{'whole milk'}
61.	{'newspapers'}	62.	

(b) FP-conditional trees whose height is larger than 1:

Note: The item name above each tree graph shows which item the FP-conditional tree belongs to.

1.	"whipped/sour cream"  ['Null Set 1', ['whole milk 317']]
2.	"citrus fruit"  ['Null Set 1', ['whole milk 300']]
3.	"pastry"  ['Null Set 1', ['whole milk 327']]
4.	"sausage"  ['Null Set 1', ['rolls/buns 301']]
5.	"tropical fruit"  ['Null Set 1', ['whole milk 416', ['other vegetables 168']], ['other vegetables 185']]

6.	"root vegetables"  ['Null Set 1', ['whole milk 481', ['other vegetables 228']], ['other vegetables 238']]
7.	"bottled water"  ['Null Set 1', ['whole milk 338']]
8.	"yogurt"  ['Null Set 1', ['whole milk 551', ['other vegetables 219', ['rolls/buns 59']], ['rolls/buns 94']], ['other vegetables 208', ['rolls/buns 54']], ['rolls/buns 131']]

9.	 ['Null Set 1', ['whole milk 394', ['other vegetables 94'], ['rolls/buns 87', ['other vegetables 43']]], ['other vegetables 131'], ['rolls/buns 290',['other vegetables 54']]]
10.	 ['Null Set 1', ['whole milk 557', ['other vegetables 176']], ['other vegetables 243']]
11.	 ['Null Set 1', ['whole milk 736']]