File: homework2.pdf
Author: P.J. Leyden
Date: October 1st, 2019

Homework 2

1. Code is in zip marked ‘myapriori.zip’. To run, use the ‘make’ command.
    1. The pseudo code:

```
File: apriori_pseudo_code.txt
Author: P.J. Leyden
Date: October 1st, 2019

Apriori Algorithm Pseudo Code

//Preconditions
Let ck represent the series of subsequent candidate sets where k = length of the contained candidate sets.
Let fk represent the series of subsequent frequent sets where k = length of the contained candidate sets.
Let 'sets' represent the set of all itemsets
Let min_sup represent the minimum support
Let results represent the final set ofall frequent item sets

//Generate the initial candidate set
for each set in sets
        for each item in set
                if item is unique
                        add to ck
max = ck.size

//main loop
for cur_szie = 1; cur_size <= max; ++cur_size

        //create map of frequency
        map ck_freq
        for each set in c1
                ck_freq.push_back(set, 0)

        //iterate through the candidate set and check for frequency against database
        for ck_itr = ck.begin; ck_itr != ck.end; ++ck_itr
                for each set in sets
                        i1 = ck_itr->begin
                        i2 = set.begin
                        while i2 != set.end AND i1 != ck_itr->end
                                if(i1 == i2)
                                        ++i1
                                ++i2
                        if i1 == ck_itr->end
                                ck_freq.find(*ck_itr).second++

        //determine the freqent sets and add them to the final result
        for each pair in ck_freq
                if pair.second >= min_sup
                        fk.add(pair.first)

        //add frequent sets to results
        for each set in fk
                results.add(set)
```

```
        //generate c(k+1)
        ck.clear
        ck = generate_next_candidate_set(fk, f1)



//candidate gen function
set<set> generate_next_candidate_set(f(k-1), f1)

        //Preconditions
        Let result be the resultant set of sets
        Let cur_set be the current set

        //Algorithm
        k = f(k-1).begin.size + 1

        for each set in f(k-1)
                for each element in f1
                        if set.find(element) == se.end
                                set_clone = set
                                for each element2 in set_clone
                                        if      element < element2
                                                set_clone.insert(element)
                                                if result.find(set_clone) == result.end
                                                        result.add(set_clone)
                                                break

        return result
```

Table 1 - Also in the file apriori_pseudo_code.txt in the zip file

2. Screen Captures of running code.

1. Minimum Support 10%

```
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$ ./myapriori.out datafile1.txt 10
Getting Minimum Support
Min Support: 10
Reading Data Sets
Running Apriori Algorithm
Completed Apriori Algorithm
Printing Results
Frequent Item Sets:
====================================
1
2
3
4
5
6
1 2
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
4 5
4 6
5 6
1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6
1 2 3 4
1 2 3 5
1 2 3 6
1 2 4 5
1 2 4 6
1 2 5 6
1 3 4 5
1 4 5 6
2 3 4 5
2 3 4 6
2 4 5 6
3 4 5 6
1 2 3 4 5
1 2 4 5 6
====================================
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$
```

2. Minimum Support 20%

```
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$ ./myapriori.out datafile1.txt 20
Getting Minimum Support
Min Support: 20
Reading Data Sets
Running Apriori Algorithm
Completed Apriori Algorithm
Printing Results
Frequent Item Sets:
====================================
1
2
3
4
5
6
1 2
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
4 5
4 6
5 6
1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 4 5
1 4 6
2 3 4
2 3 5
2 3 6
2 4 6
3 4 5
4 5 6
1 2 3 4
1 2 3 5
====================================
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$
```

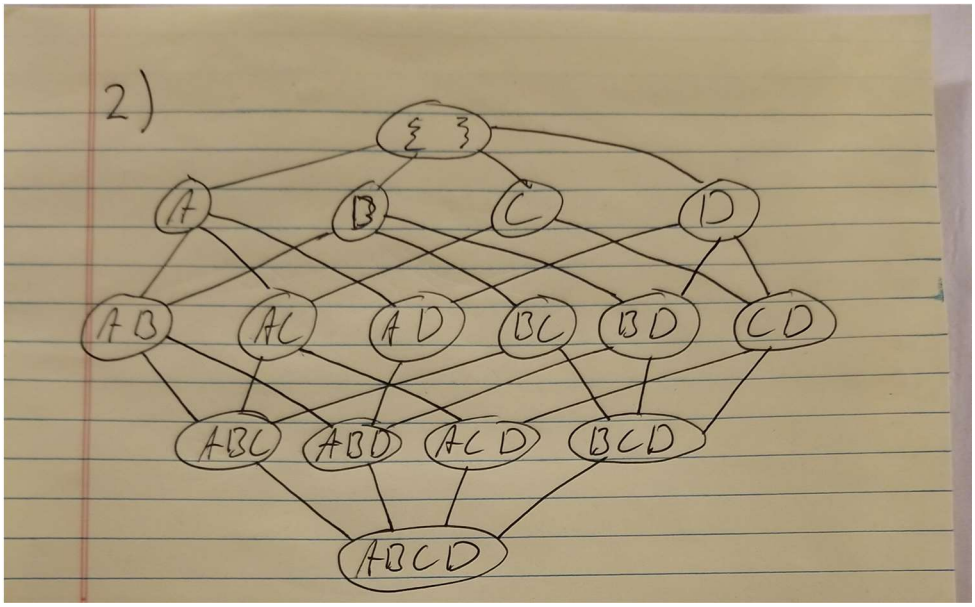
3. Minimum Support 30%

```
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$ ./myapriori.out datafile1.txt 30
Getting Minimum Support
Min Support: 30
Reading Data Sets
Running Apriori Algorithm
Completed Apriori Algorithm
Printing Results
Frequent Item Sets:
====================================
1
2
3
4
5
6
1 2
1 3
1 4
1 5
2 3
2 4
2 6
3 4
3 5
4 5
4 6
1 2 3
1 2 4
2 3 4
====================================
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$
```

4. Minimum Support 50%

```
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$ ./myapriori.out datafile1.txt 50
Getting Minimum Support
Min Support: 50
Reading Data Sets
Running Apriori Algorithm
Completed Apriori Algorithm
Printing Results
Frequent Item Sets:
====================================
1
2
3
4
5
1 2
2 3
2 4
3 4
====================================
wolfdragoon31@ubuntu:~/Programming/cs_projects/data_mining/hw2/apriori_implementation$
```

2. Question 2



3. Question 3
    1. With min support 7. The frequent item-sets are:
        1. {A} with support, 8
        2. {B} with support, 7

4. Answer to the Bonus Question: