- Dataset is from IBM Quest Synthetic Data Generator
- Three methods were used in this project: **brutal force**, **Apriori algorithm** and **FP-growth**

1. Test with different size of data (different number of transaction)

   Table 1 shows executing time of the three methods with four different size of data (10, 100, 1k, 10k and 100k transactions). Here the **minimum support** was set as **0.6**

   Table 1 Execution time of methods with different number of transactions (sec)

	Brutal force	Apriori algorithm	FP-growth
10 transactions	0.13	0.003	0.001
100 transactions	0.2	0.004	0.002
431 transactions	0.2	0.01	0.003
1k transactions	x	0.11	0.045
10k transactions	x	0.99	0.43
100k transactions	x	3.4	1.65

   For small data, say couple hundreds transactions and below, the execution time of Apriori algorithm and FP-growth are **close**. Sometimes, that of Apriori may even be faster, and both will sometimes output zero sec. For very small data, say about ten transactions, time for executing brutal force method is also small, though longer than those of Apriori and FP-growth. However, for transactions more than **431**, my brutal force was not able to run the results because of too many items need to be assembled for the later scan in dataset, making the execution **very slow**, see the "x" symbol of brutal force.

   For large data, say thousand to tens of thousands or even hundred thousand transactions, the FP-growth can be about **faster** than Apriori **by more than half**.

2. Correctness of FP-growth and Apriori algorithm

   Initially, I used the example in professor's lecture note (that one has beer and bread) to construct and check my code of brutal force, Apriori and FP-growth. When the size of transaction is small, the check with frequent itemsets can be done, but as the data size increases, the length of frequent itemsets is more convenient for checking. See table 2 for the number of frequent itemsets by the three methods. Note that the number of frequent itemsets from the three methods are supposed to be the same, but for small data like one hundred and ten transactions, those by FP-growth are a bit less than those from brutal force and Apriori, see the red values of FP-growth. I have tried with very small data to check the correspondence of the frequent itemsets, but fail to fix the difference. However, as the size of data increases, the results of frequent itemsets number are the same for Apriori and FP-growth.

Table 2 Number of frequent itemsets by methods with different number of transactions (min.Supp = 0.6)

	Brutal force	Apriori algorithm	FP-growth
10 transactions	159	159	134
100 transactions	65	65	62
431 transactions	69	69	69
1k transactions	x	204	204
10k transactions	x	195	195
100k transactions	x	53	53

3. Test of same dataset with different setting of minimum support

   Here I try to adjust the value of minimum support to see the change of frequent itemsets, data of 10k transactions is used here. The increasing speed of frequent itemsets as higher minimum support can be seen from table 3. Also, as the threshold being lower, the **execution time** of both methods **increase a lot**.

Table 3 Number of frequent itemsets by methods with different value of minimum support (10k transactions)

	Apriori algorithm	FP-growth
min.Supp = 0.9	2	2
min.Supp = 0.8	17	17
min.Supp = 0.7	57	57
min.Supp = 0.6	195	195
min.Supp = 0.5	677	677
min.Supp = 0.4	2614	2614
min.Supp = 0.3	11794	11794

4. Discussion

   - The limitation of brutal force (maybe just mine) comes from the next level generation of candidate itemsets, and I found that if the number of variables in IBM Quest Synthetic Data Generator (-nitems) are more than 10, the process of candidate generation (refer to fig. 1) will be slow enough for me to stop the execution.

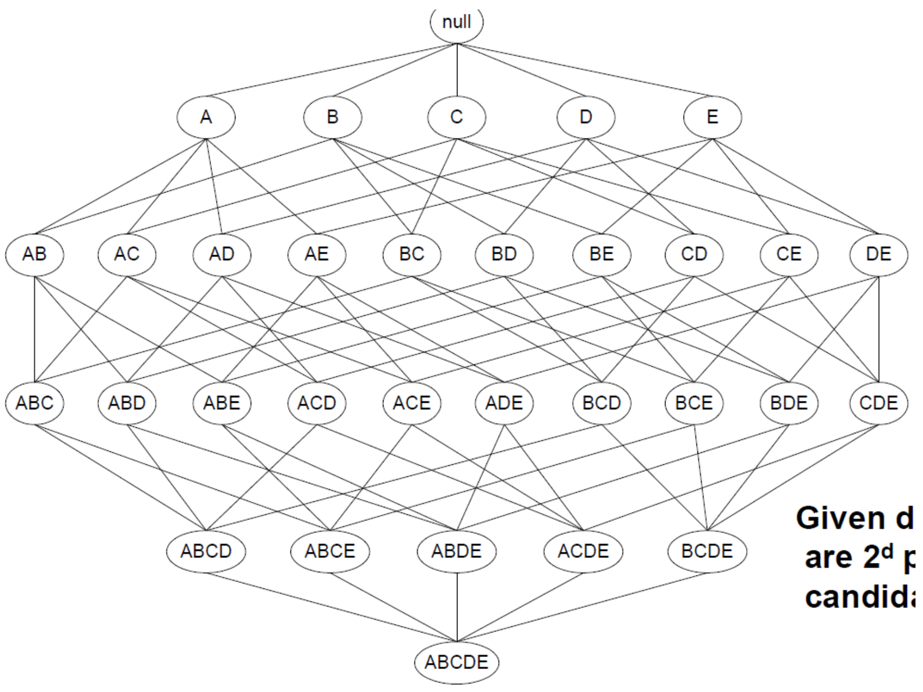

Fig. 1 the example of possible candidate itemsets

- Apriori algorithm improve the candidate generation process in the brutal force (or the original) method, it prunes the itemsets which are found to be infrequent in advance, and thus the latter number of candidate itemsets are reduced a lot, see fig. 2.

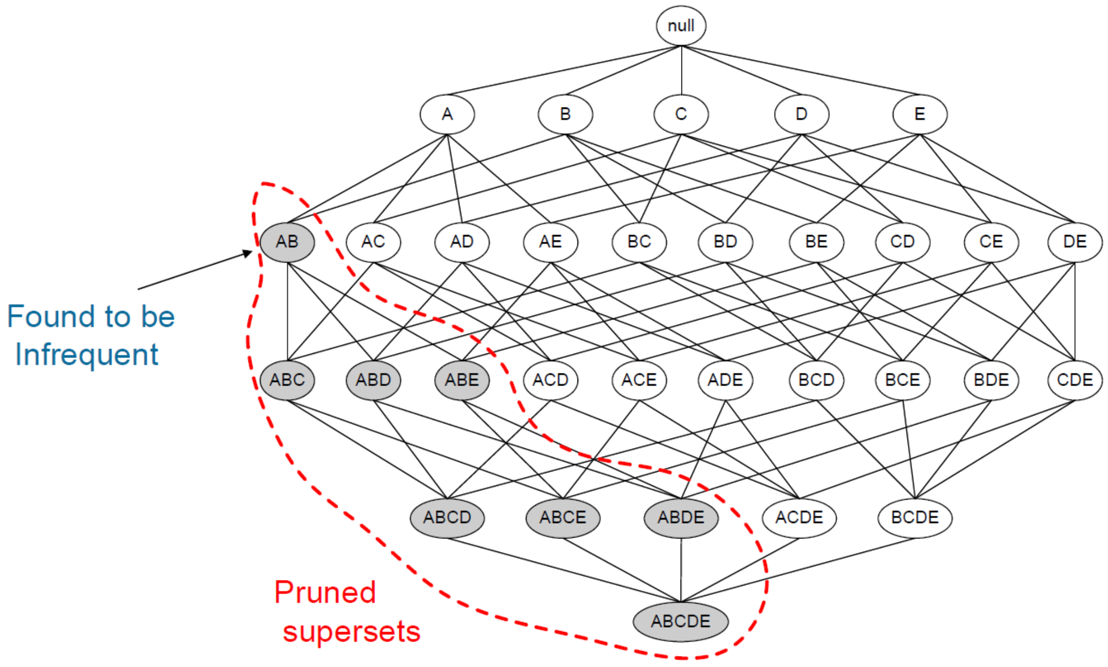

Fig. 2 The core idea of Apriori algorithm

- Without the candidate generation and test process, FP-growth is even strong in efficiency, but the tradeoff is the complexity of construction. Also, the mining based on each conditional database is much less computationally loaded than the original large dataset.