

Data Mining
Prof. Pabitra Mitra
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 05
Apriori Algorithm

Welcome to association rules part 2. So, as we had discussed what we want to do is to find out all association rules of the form $X \rightarrow Y$ where X is an item set and Y is an item set such that the support and confidence criteria are satisfied support and confidence criteria are satisfied.

(Refer Slide Time: 00:50)

- Association Rule
 - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
 - Example:
 $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$
- Rule Evaluation Metrics
 - Support (s)
 - ◆ Fraction of transactions that contain both X and Y
 - Confidence (c)
 - ◆ Measures how often items in Y appear in transactions that contain X

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example:

$$\{\text{Milk, Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

So, we have defined support to be the number of transactions where the union X and Y appear and confidence to be the fraction of transactions. So, where Y appear as well Y as well as X appear divided by number of transactions.

(Refer Slide Time: 01:13)



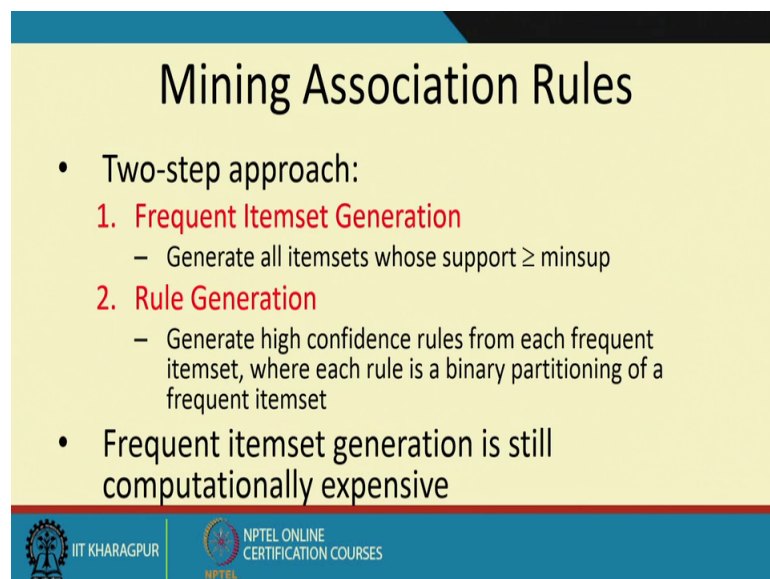
Association Rule Mining Task

- Given a set of transactions T , the goal of association rule mining is to find all rules having
 - support \geq *minsup* threshold
 - confidence \geq *minconf* threshold
- Brute-force approach:
 - List all possible association rules
 - Compute the support and confidence for each rule
 - Prune rules that fail the *minsup* and *minconf* thresholds \Rightarrow **Computationally prohibitive!**

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES



Where only X appear both the standard ratios are greater than some preset threshold *minsup* and *minconf* we call it to be a association rule and we find out that if we try to discover such rules finding all possible combination is computationally prohibitive.

(Refer Slide Time: 01:31)



Mining Association Rules

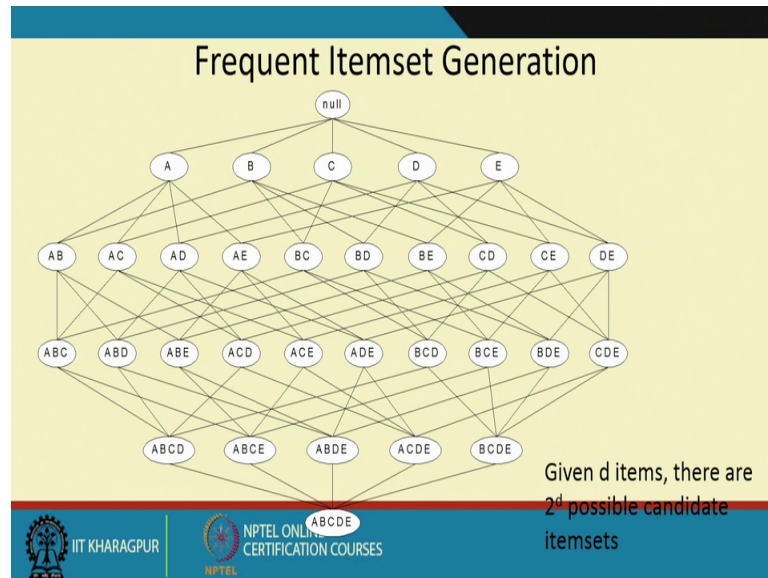
- Two-step approach:
 1. **Frequent Itemset Generation**
 - Generate all itemsets whose support \geq *minsup*
 2. **Rule Generation**
 - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

So, then we adopt a 2 step approach the first approach is find out all item sets all sets of item whose support is greater than *minsup* and for each of these item sets we divide them into 2 parts 2 sets of item one in the left one in the right and then for each of these partitions we see we check the confidence if it is greater than the confidence threshold

we select it. So, there are 2 sets; 2 steps, first is finding frequent items set second is doing the rule generation from the identified frequent item sets.

(Refer Slide Time: 02:28)



So, but still this finding frequent item set is computationally expensive because let us see why it is computationally expensive. So, suppose I have 5 items in a store; for example, let me denote them as A, B, C, D and E; beer, coke, diaper, bread, milk, E E A B A to E; 5 items.

So, I can form a item set of sides 5; 5 item set, you remember the definition of k items set 5 items set sizes 5 all A B C D; one such 5 item sets are there, there are 4 possible 4 item sets containing 4 elements A B C, D A B, C A B, D and so on, there are 10 possible 3 item sets A B C, A B D and so on. So, basically what we are doing is that see a person suppose there are 5 items available for sale in the store a customer can buy any of these combinations a customer can buy all 5 items a customer can buy 4 out of these 5 items a customer can buy 3 out of this 4-5 items, 2 out of this 5 and so on. So, these; all these ellipses white ellipses shows that and then finally, a null a customer might not buy anything. So, this lattice sort of tells us that all the possible item sets that might be present in a market basket.

We call them as candidate item sets in any choice of purchases. So, if there are d possible items you can easily check that there are 2 to the power d possible item sets possible exponential to depart d possible items sets possible. So, either one item set is present or

absent in a in a bag. So, 5 are there; d are there. So, to the power d not only that you see that these item sets you can sort of arrange them in a lattice like structure. So, in the top of the lattice is no item set bottom of the lattice is a 5 item set or k items set and then at each level of the lattice there are k; k minus 1 k minus 2 and so on.

So, 4 item sets 3 item sets 2 item sets one item sets and. So, on and then I can also draw a line connecting this individual level saying that if one is a subset of the other for example, the set A B C D; 4 items set A B C D can produce 4 different subsets of size 3 A B C, A B D, A B E, A C D, B C D. So, if I look at this there are sorry there are this this will produce this all right just a minute. So, there are this A B C D, it will produce a B C as a sub. So, I draw this line if say this if this is a subset of this then I draw a line. So, for example, a B C is a subset of A B C D, A B D is a subset of A B C D and so on. So, if I arrange this 2 to the power d subsets I can sort of draw this kind of a lattice saying that which is a subset of which is a subset of it.

If you carefully see this diagram you will see that these lines are nothing, but one is a subset of and this can go across all levels for example, this A B C is subset are A B, A C and B C, A B is a subset of A B subsets are A and B and so on. So, the thing to be noted is that this lattice represents all possible candidate item sets that one may think of now tell me; how do I find. So, my first job is to find out what are the frequent item sets M.

(Refer Slide Time: 07:50)

Frequent Itemset Generation

- Brute-force approach:
 - Each itemset in the lattice is a **candidate** frequent itemset
 - Count the support of each candidate by scanning the database

Transactions


TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

N (rows), w (width)


List of Candidates

M (rows)

– Match each transaction against every candidate
 – Complexity $\sim O(NMw) \Rightarrow$ **Expensive since $M = 2^d$!!!**



IIT KHARAGPUR



NPTEL ONLINE
CERTIFICATION COURSES

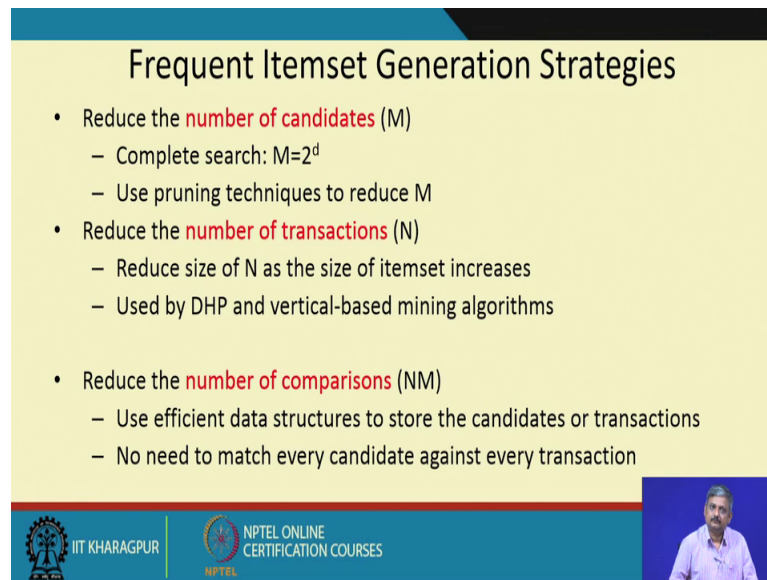
So, how I do that 5 set a support threshold you know frequent means number of times is more than the support I said a suppose threshold maybe I said a suppose threshold that it is 10 percent of the total number of transactions. So, if the total number of transaction is one hundred a particular item set should appear more than 10 times more than 10 times; it should appear. So, what I will do to find out what are the; if that is.

So, then I call it as a frequent item set. So, how do I do it? So, I will take any of this for example, I will take A B C D, I will go through this hundred transactions and see whether each of them contains A B C D or does not contain A B C D, I will scan this entire hundred transaction find out if it contains A B C D or does not and then I will count how many of this hundred are containing A B C D, if that number is greater than this 10 percent of the total then I will say this A B C D is a frequent size of set I will say A B C D is a frequent item set, right. So, to among these all possibilities of different item sets if by brute force I want to find out who among them are frequent each and every one of these for each and every one of these I will scan; the scan the set of transactions hundred transaction.

In fact, there will be millions 100 transactions I will check how many times they appear I will check if the cross the threshold and I will find out who among them not all some of them among these candidates all these are candidates are indeed frequent are indeed frequent; that means, they appear more than means of number of times. So, what is the complexity of this if the total number of transaction item sets possible are m which in this case is 2 to the power D and total number of elements in the transaction in the database that is number of n number of transactions is n and the average length of each w because we have to match the string is w then it is n into m into w which since m is large number 2 to the power D n is also large.

So, this indeed will be very high ok.

(Refer Slide Time: 11:03)



Frequent Itemset Generation Strategies

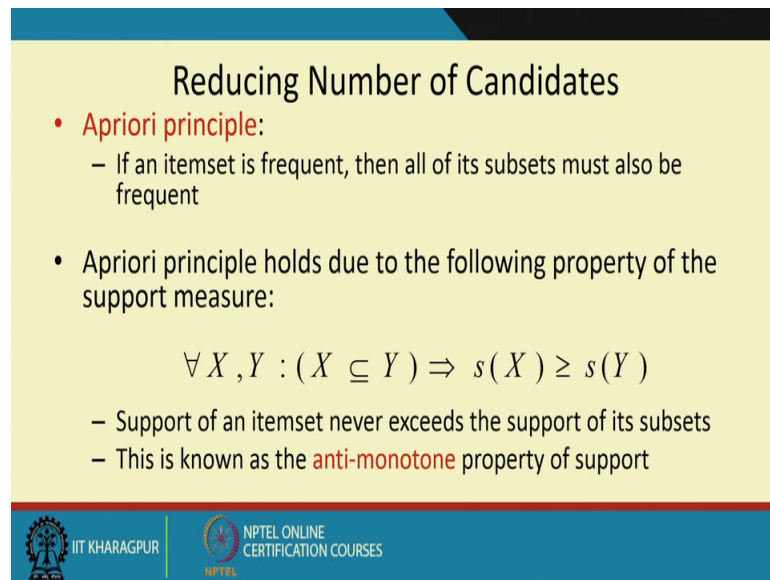
- Reduce the **number of candidates** (M)
 - Complete search: $M=2^d$
 - Use pruning techniques to reduce M
- Reduce the **number of transactions** (N)
 - Reduce size of N as the size of itemset increases
 - Used by DHP and vertical-based mining algorithms
- Reduce the **number of comparisons** (NM)
 - Use efficient data structures to store the candidates or transactions
 - No need to match every candidate against every transaction

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, now the thing is that so, but how can we reduce this time let me give you an intuition the intuition is that do I really need to check each and every possible candidate item sets to check if they are supporting if they are satisfying the support condition or if I know somehow that one of these is not frequent one of these is not frequent can that help me to figure out whether these subsets of that will be frequent or not frequent. So, in other words if I know these to be a frequent item set they appear a large number of times can I say anything about this thing let me tell this in plain English; the English is if I know that people buy people do not buy say milk.



And bread a lot of times will do not buy milk and bread a lot of times; that means, the set milk and bread is not frequent can I say what can I say about will people buy milk and bread and sugar frequently. So, if I already know that people do not buy milk and sugar frequently or let me give another example if I already know people do not buy cricket bat and cricket ball frequently I can automatically infer that people definitely do not buy cricket bat cricket ball and cricket gloves frequently. So, in other words if cricket bat and ball is not a frequent item set cricket bat and ball and gloves is definitely not a frequent item set is definitely not; that means, I will explain soon why it is. So, and; that means, if A B C; I know not to be a frequent item set I know A B C D is definitely not a frequent item set.

(Refer Slide Time: 13:48)



Reducing Number of Candidates

- **Apriori principle:**
 - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$
 - Support of an itemset never exceeds the support of its subsets
 - This is known as the **anti-monotone** property of support

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES



So, let me let me state this more clearly I can use this. So, this this principle is known as the Apriori principle what this principle says is that is actually another way of putting what I said if some item set is frequent then its subset should also be frequent and a way of setting this as if people buy milk and bread a lot of time definitely it is obvious that people also buy milk a lot of time only milk a lot of time. So, it says that if X is a subset of Y the support of X is definitely going to be more than the support of Y it is definitely more going to be more than the support of Y. In other words, the support of an item set never exceeds the support of this subset let me again explain this with an example suppose I consider this set or let me take this set sorry.

(Refer Slide Time: 15:17)

Definition: Frequent Itemset

- **Itemset**
 - A collection of one or more items
 - Example: {Milk, Bread, Diaper}
 - k-itemset
 - An itemset that contains k items
- **Support count (σ)**
 - Frequency of occurrence of an itemset
 - E.g. $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$
- **Support**
 - Fraction of transactions that contain an itemset
 - E.g. $s(\{\text{Milk, Bread, Diaper}\}) = 2/5$
- **Frequent Itemset**
 - An itemset whose support is greater than or equal to a *minsup* threshold

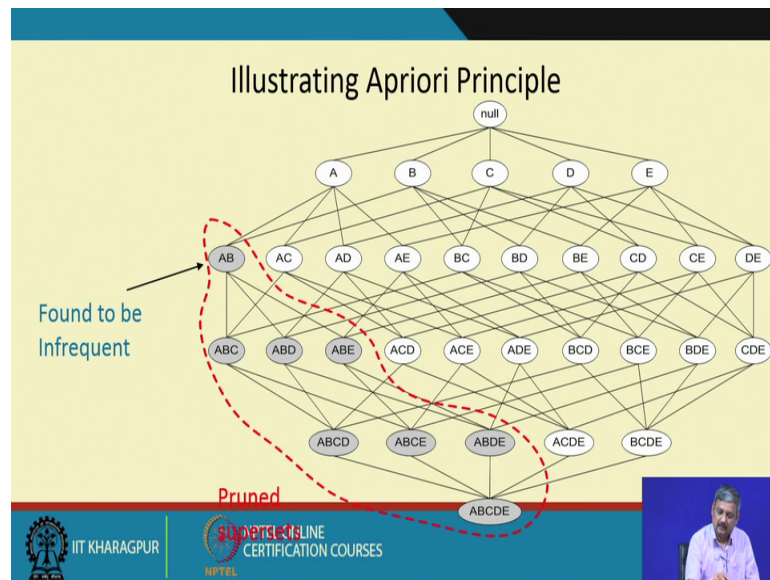
<i>TID</i>	<i>Items</i>
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke



Let me consider this set, suppose I consider the item set bread and milk and diaper, bread and milk and diaper; how many times it appears it appears 3 times, no 3 time, 2 times bread and milk and diaper appears 2 times row, one time here and one time here, the set bread and milk and diaper now. So, 2 times it appears. So, the support is 2, now you take a subset of these 3 say only bread and milk, only bread and milk; one subset. So, how many times it appear? Definitely, it should appear either 2 times or more than 2 times. In this case, it appears 3 times bread and milk only. So, take another subset take the set bread and diaper bread and diaper. So, how many times it appear it appears 3 times. So, this actually is in agreement with this property that this property that if X is a subset of Y a smaller set it.

Support should be greater than the support the super sets; that means, smaller item sets subset item sets have more support than their super sets. So, this is actually known as the anti monotone property anti monotone property of the support function. So, you can directly obtain the Apriori principle from this anti monotone property, it is an obvious extension that it naturally means that if we apply some threshold if an item set is frequent its subsets must also be frequent.

(Refer Slide Time: 17:58)



So, this property actually helps us avoid evaluating the frequency of all these possible candidate item sets why because I have already said that if a if a item set is frequent only subsets would be frequent other way around if something; some items it is not frequent none of its super sets can be frequent.

So, basically if A B is frequent not frequent sorry if A B is not frequent, A B C cannot be frequent, A B D cannot be frequent, A B E cannot be frequent, if A B C is not frequent, A B C D cannot be frequent A B C cannot be frequent, if A B C D is not frequent, A B C D E cannot be frequent; that means, knowing that this particular set of items A B is not frequent can immediately infer that none of this can be frequent none of this can be frequent. So, how this helps this helps in candidate item set pruning earlier everybody was a candidate for being a frequent item set now moment I examine one I can remove this from my candidate set I can I need to examine all of this again I will examine some of them if they are found to be infrequent they are entire parent set will be pruned ok.

(Refer Slide Time: 19:51)

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk,Diaper}	3

Triplets (3-itemsets)

Minimum Support = 3

If every subset is considered,
 ${}^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With support-based pruning,
 $6 + 6 + 1 = 13$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, that is the principle enter. So, here I illustrate it with an example suppose there are this six items bread coke milk beer diapers and these are their counts and since there are six items all possible combination is 41. If I set some support threshold to be greater than 2 that is 3 and above, then only 13 of this 41 are frequent instead of evaluating all these 41; what I can do is that I can remove this coke; an egg from further configurations take only bread milk and diaper find out 2 item sets combining bread, milk, beer, diaper, only find their count find 2 of them are infrequent take only this other and get one sub size 3 and so on.

(Refer Slide Time: 21:09)

Apriori Algorithm

- Method:
 - Let $k=1$
 - Generate frequent itemsets of length 1
 - Repeat until no new frequent itemsets are identified
 - Generate length $(k+1)$ candidate itemsets from length k frequent itemsets
 - Prune candidate itemsets containing subsets of length k that are infrequent
 - Count the support of each candidate by scanning the DB
 - Eliminate candidates that are infrequent, leaving only those that are frequent

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, this way I can proceed. So, this forms the basis of the frequent item set finding algorithm known as the Apriori algorithm; how does the algorithm proceed you first generate frequent item sets of length one of length one; one item sets find out how many of them are frequent. So, if we look at this picture if I look at this picture what I do first I individually find out A B C D E which of these are frequent as a single item and mark only those which are frequent prune out the remaining lattice among the ones that stay frequent I find out all their combinations in the next level 2 items at level. So, I check which of them are frequent by counting and then if somebody is not frequent they are further sub it I edit, I prune and then I go on till I exhaust go till the bottom and then I will find out.

So, the steps are this generate item sets of length n length one and then recursively from frequent item sets of length k from item sets of length k plus one why how you can see that if I know which are 2 item sets are which are frequent from that I find 3 item sets which are frequent k; k plus 1, of course, to find out whether their frequent or not you have to scan the entire database of transactions count and do it.

(Refer Slide Time: 23:44)

Factors Affecting Complexity

- Choice of minimum support threshold
 - lowering support threshold results in more frequent itemsets
 - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
 - more space is needed to store support count of each item
 - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
 - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
 - transaction width increases with denser data sets
 - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

 IIT KHARAGPUR
  NPTEL ONLINE CERTIFICATION COURSES

So, of course, these are the factors which will still effect your complexity how many what is the support threshold; basically how much pruning you can do number of items in each transaction size of the database n transaction width. So, all this will effect, but still what will happen is that you no longer evaluate all these 2 to the power d item sets,

many of them will be pruned in this process many of them will be pruned in this process. So, let me quickly give an example of how these proceeds.

So, you can actually visualize it with this example. So, if we consider six item sets. So, all so many different combinations are possible and suppose I set my k for the first item set I count what is the support apply the threshold find out frequent one item set frequent one item set from the one item sets I note that I this in the 2 item sets this ruled out black ones which are infrequent coke and they do not appear only the remaining.

Ones I combine; I combine to get these 2 or 2 item sets note that I have to again count for this too it is not that from this count I can from this count I can find out this count I have taken count here go to the database and count and find out which are frequent and again I from that I find that 3 item sets. So, this way I proceed this will go on till either you find the entire number of item sets or no new adaptive lattice is pruned no new can be all right.

So, this definitely reduces the complexity and this is applied in can be applied in last databases, in the next class, we will see after finding out the frequent item sets, how I will form rules from them using a similar Apriori, I like framework and we will see from the rules how we can evaluate them and do them there are many other algorithms which further reduce the complexity like FPT frequent pattern tree we will discuss them in the subsequent lectures.

Thank you for today.