Apriori Algorithm

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns
- The disadvantage of Apriori algorithm are:
  - It may need to generate a huge number of candidate sets.
  - It may need to repeatedly scan the database and check a large set of candidates by pattern matching.
  - Breadth-first search approach

FP-growth Algorithm

- Frequent Pattern growth (FP-growth) Algorithm was proposed by J. Han, J. Pei, and Y. Yin.
- Mining frequent itemsets without candidate generation. FP-growth adopts depth-first search approach.
- The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies).
- Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count.

FP-growth Algorithm

- An FP-tree is then constructed as follows. First, create the root of the tree, labeled with "null." Scan database D a second time.

- The items in each transaction are processed according to descending support count order and a branch is created for each transaction.

Table 4

Dataset *D*

TID	List of items
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2, I3, I5
9	I1, I2, I3

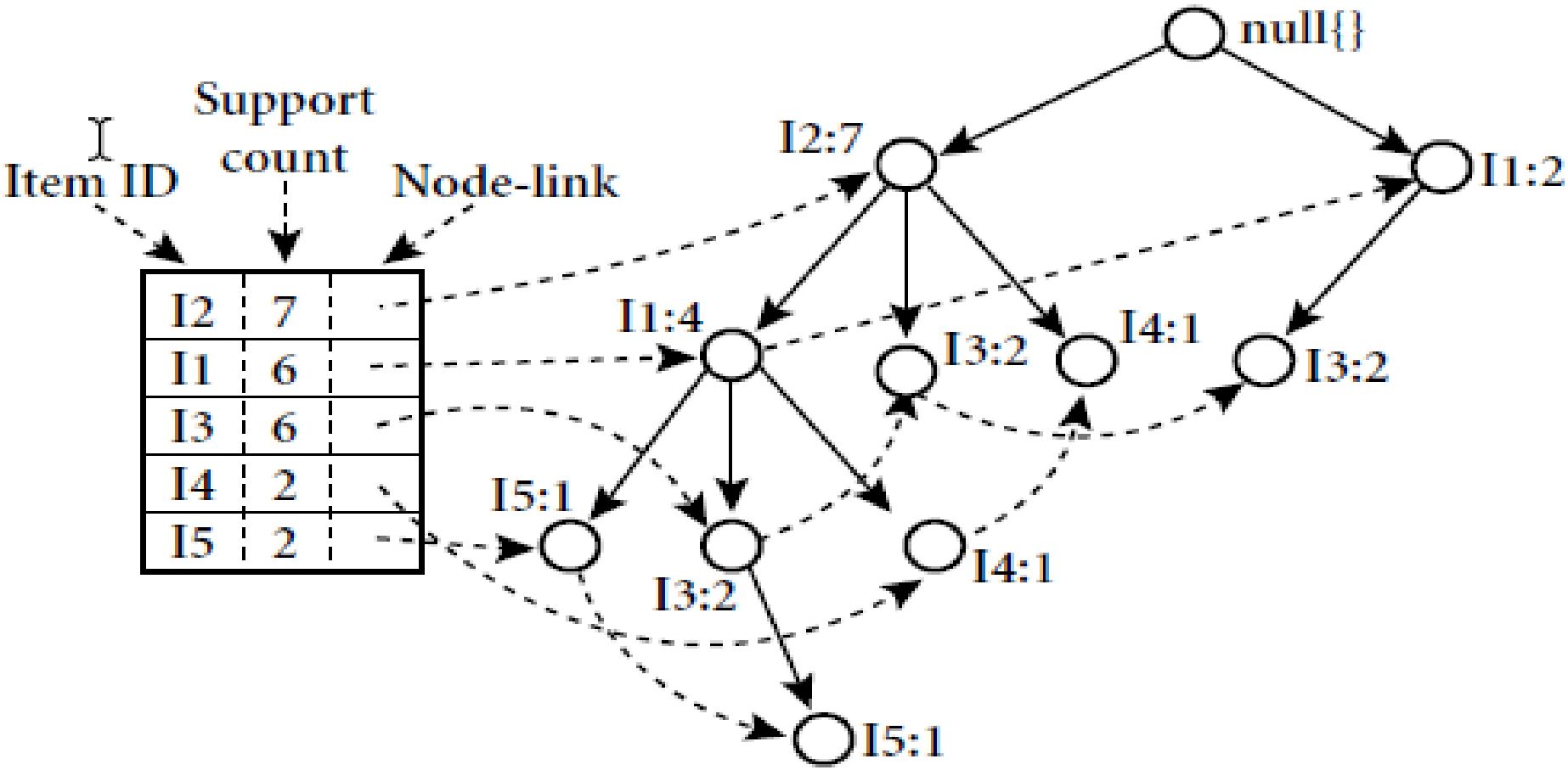
FP-growth Algorithm

- The set of frequent items is sorted in the order of descending support count. Thus, we have {{I2: 7}, {I1: 6}, {I3: 6}, {I4: 2}, {I5: 2}}.
- The items in each transaction are processed according to descending support count order and a branch is created for each transaction.

Table 5
Dataset D

TID	List of items
1	I2, I1, I5
2	I2, I4
3	I2, I3
4	I2, I1, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I2, I1, I3, I5
9	I2, I1, I3

FP-growth Algorithm



An FP-tree registers compressed, frequent pattern information.

FP-growth Algorithm

- The FP-tree is mined as follows.
- Start from each frequent length-1 pattern (as an initial suffix pattern).
- Construct its conditional pattern base (a "subdatabase," which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern).
- Then construct its (conditional) FP-tree, and perform mining recursively on such a tree.
- The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

FP-growth Algorithm

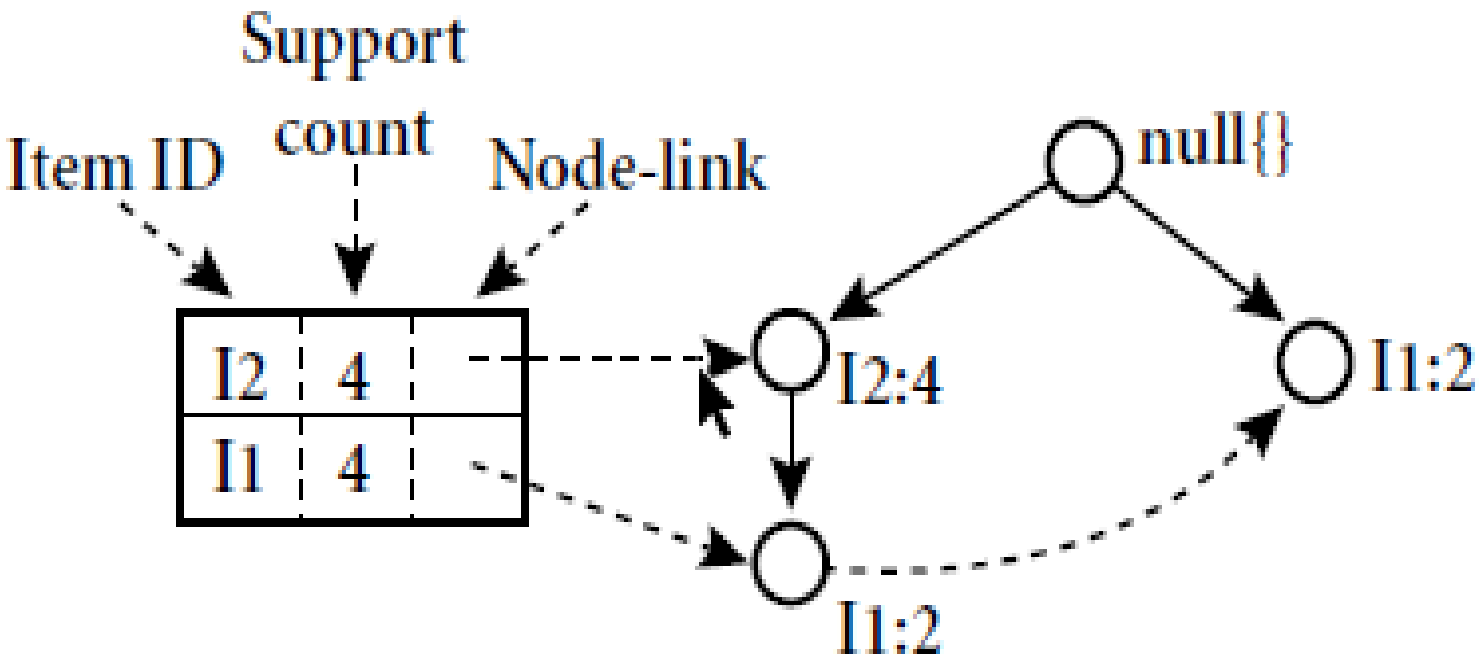
Mining the FP-tree by creating conditional (sub-)pattern bases.

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

FP-growth Algorithm

- Lets first consider I5. I5 occurs in two branches of the FP-tree.
- The paths formed by these branches are $\langle$I2, I1, I5: 1$\rangle$ and $\langle$I2, I1, I3, I5: 1$\rangle$.
- Considering I5 as a suffix, its corresponding two prefix paths are $\langle$I2, I1: 1$\rangle$ and $\langle$I2, I1, I3: 1$\rangle$, which form its conditional pattern base.
- Its conditional FP-tree contains only a single path, $\langle$I2: 2, I1: 2$\rangle$; I3 is not included because its support count of 1 is less than the minimum support count.
- The single path generates all the combinations of frequent patterns: $\{$I2, I5: 2$\}$, $\{$I1, I5: 2$\}$, $\{$I2, I1, I5: 2$\}$.

FP-growth Algorithm



The conditional FP-tree associated with the conditional node I3.

FP-growth Algorithm

- I3's conditional pattern base is $\{\{I2, I1: 2\}, \{I2: 2\}, \{I1: 2\}\}$.
- Its conditional FP-tree has two branches, $\langle I2: 4, I1: 2\rangle$ and $\langle I1: 2\rangle$
- Generates the set of patterns, $\{\{I2, I3: 4\}, \{I1, I3: 4\}, \{I2, I1, I3: 2\}\}$.