Baseline algorithm

Basic strategy[1]

- Enumeration + testing
- Using two theorem to prune

- [1]Chi Y, Xia Y, Yang Y, et al. Mining closed and maximal frequent subtrees from databases of labeled rooted trees[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(2): 190-202.

Definition

- t is a subtree, t' represents one supertree of t, i.e., t is an induced subtree of t'.
- t' has one more vertex than t.
- t'/t represent the vertex which t' has and t does not.
- $B_t = \{t' | t' \ is \ derived \ from \ t \ and \ t' is \ frequent\}$

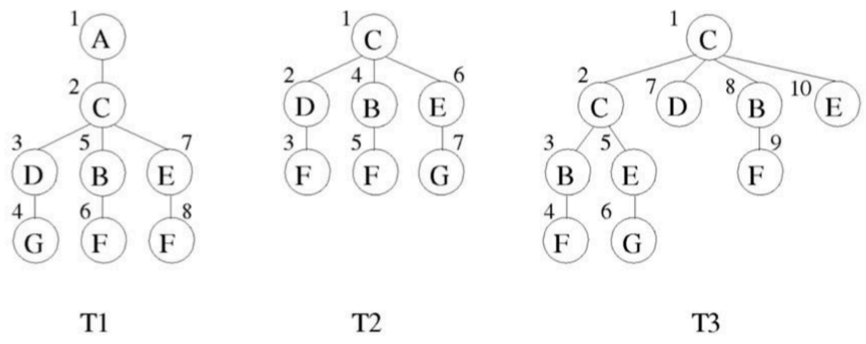
- Anti-monotonicity: if t is not frequent, then t' is impossible to be frequent.

- A subtree t is maximal $iff \ B_t = \emptyset$.

Occurrence-Matching

- For $t' \in B_t$, t' and t are ***occurrence matched*** if, for each occurrence of t in database, there is an corresponding occurrence of t'.

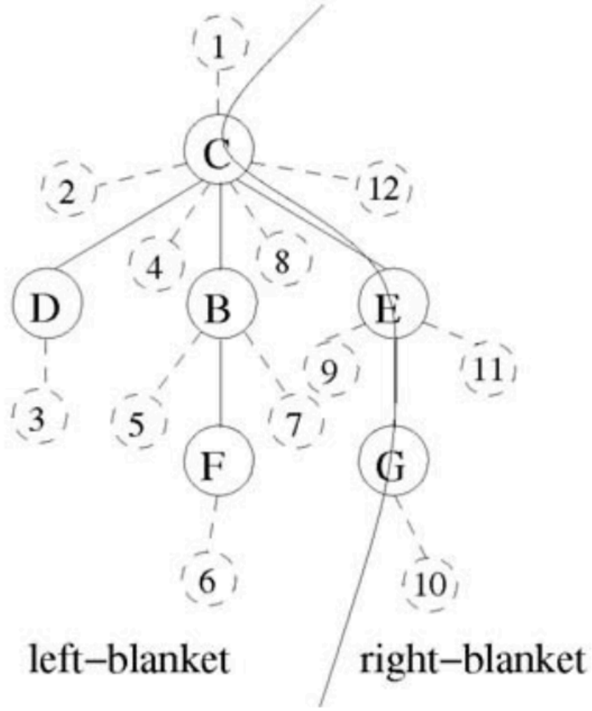
- Eg. {C-B-F} and {C-B} is occurrence matched.



Enumeration rule

- Generate new vertex in the rightmost path.
- Eg., position 10, 11, 12 are three rightmost position of the tree.
- $B_{t-right}$ = $\{t' \in B_t |\ t'/t$ is on the rightmost path of $t\}$.

- $B_{t-left} = B_t / B_{t-right}$.

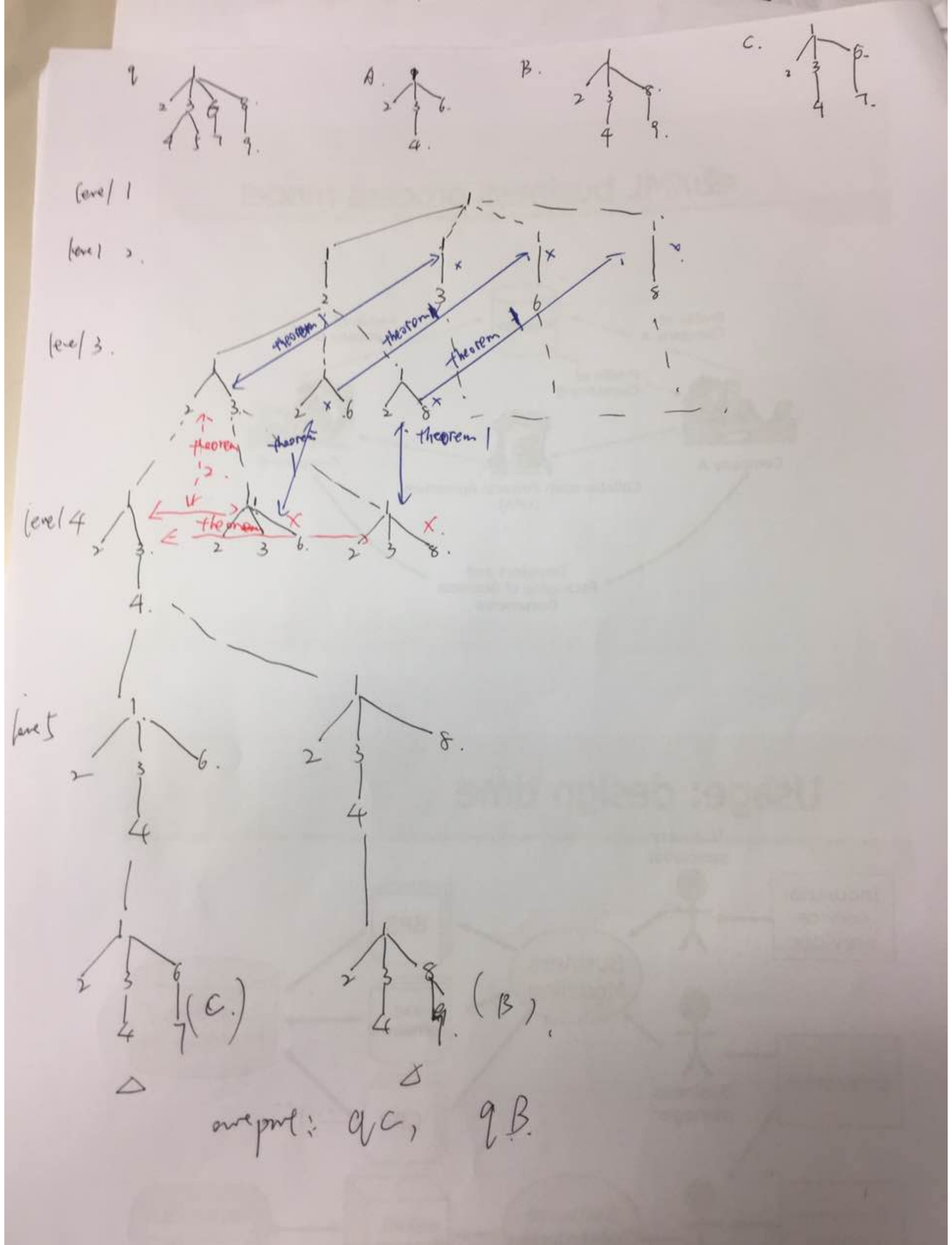


Pruning strategy

- $k - \widehat{core}$ constraint: If the number of nodes shared t is less than k, prune t.

- Theorem 1: If there exists $t' \in B_{t-left}$ such that t' and t are ***occurrence matched*** **,** then neither t nor any descents of t can be maximal. Thus we can prune t.

- Theorem 2: If there exists $t' \in B_{t-right}$ such that t' and t are ***occurrence matched*** and the parent of t'/t is v (where is v on the rightmost path of t), then we do not need to extend t by adding new rightmost vertices to any proper ancestor of v.



- Theorem 1: If there exists $t' \in B_{t-left}$ such that t' and t are ***occurrence matched ,*** then neither t nor any descents of t can be maximal. Thus we can prune t.

- Theorem 2: If there exists $t' \in B_{t-right}$ such that t' and t are ***occurrence matched*** and the parent of t'/t is v (where is v on the rightmost path of t), then we do not need to extend t by adding new rightmost vertices to any proper ancestor of v.