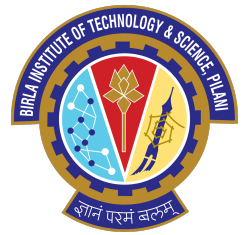
SS-ZG548: ADVANCED DATA MINING

Lecture-10: Revision

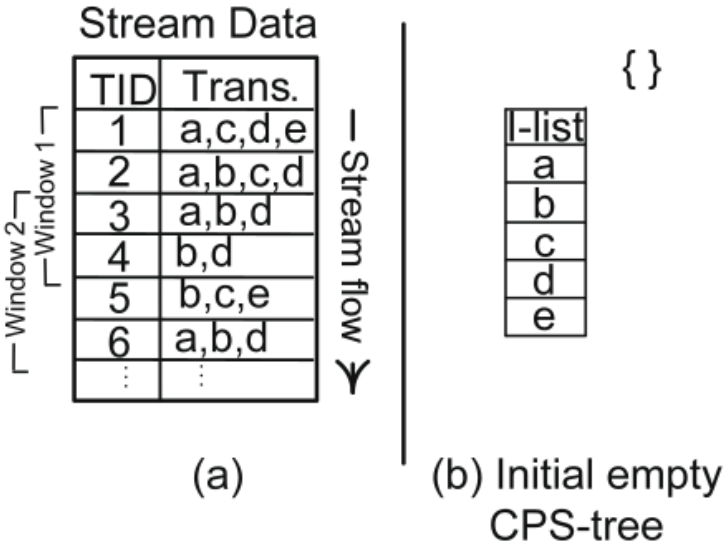


Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems Engineering,
BITS Pilani, Rajasthan-333031 INDIA

Feb 18, 2018 (WILP @ BITS-Pilani Jan-Apr 2018)

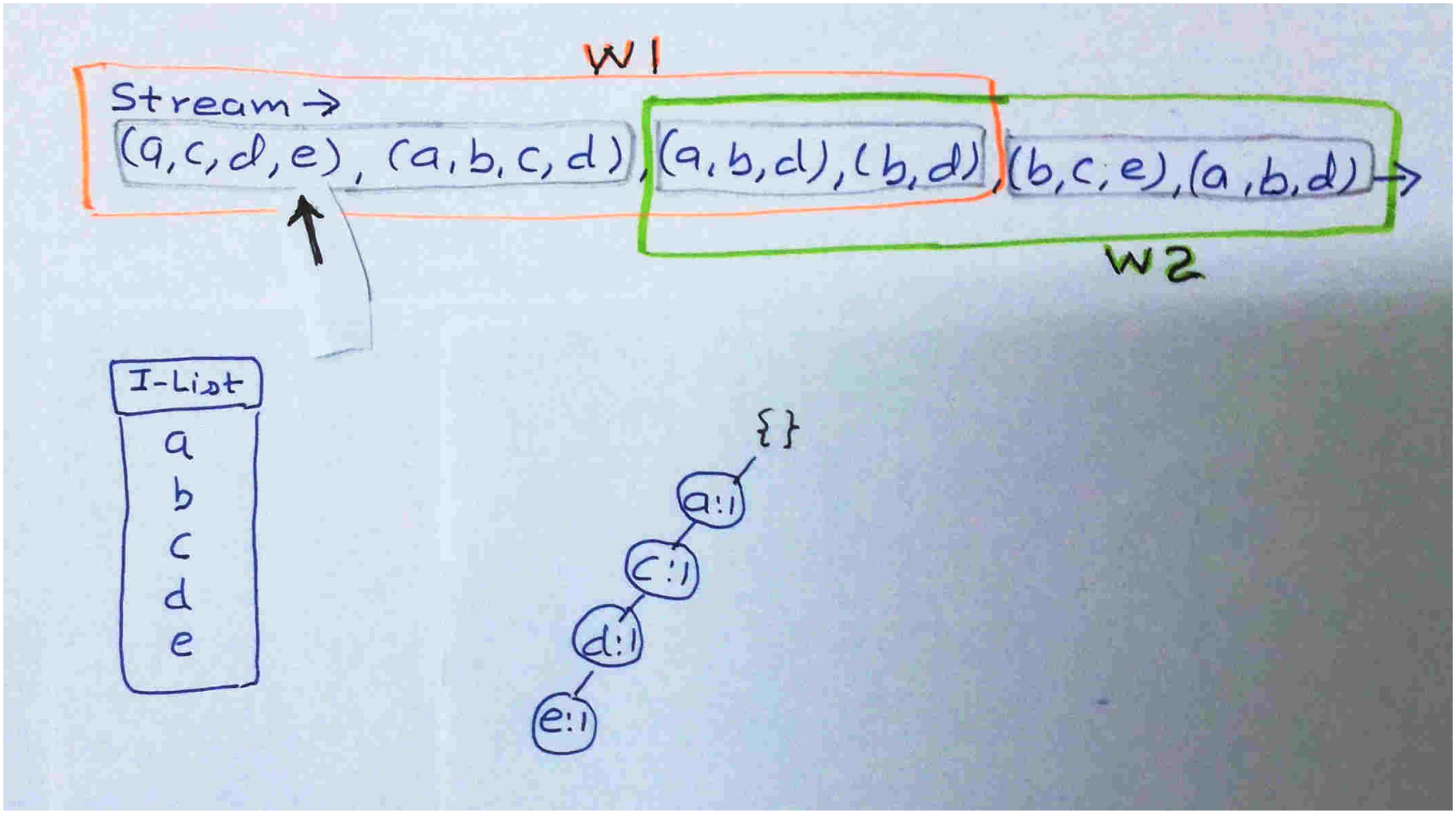
Recap: Frequent pattern mining over data streams

- Compact Pattern Stream tree (CPS-tree) [1] is a highly compact frequency-descending tree structure at runtime
- Efficient in terms of memory and time complexity
- Involves insertion and then restructuring

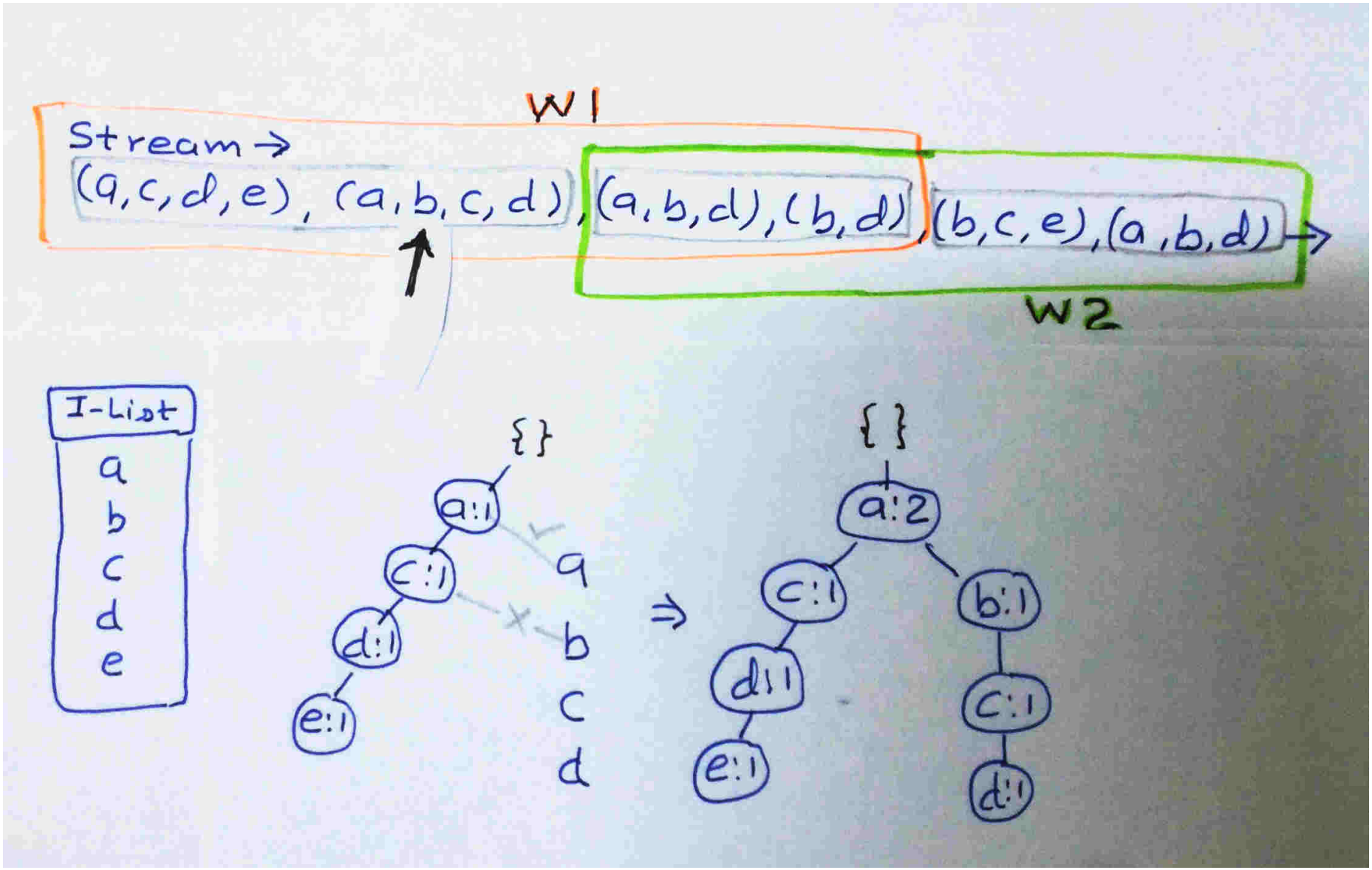


[1] Tanbeer, Syed Khairuzzaman and Ahmed, Chowdhury Farhan and Jeong, Byeong-Soo and Lee, Young-Koo, "Sliding window-based frequent pattern mining over data streams", in Information sciences, 179(22) pages 3843–3865, Elsevier, 2009

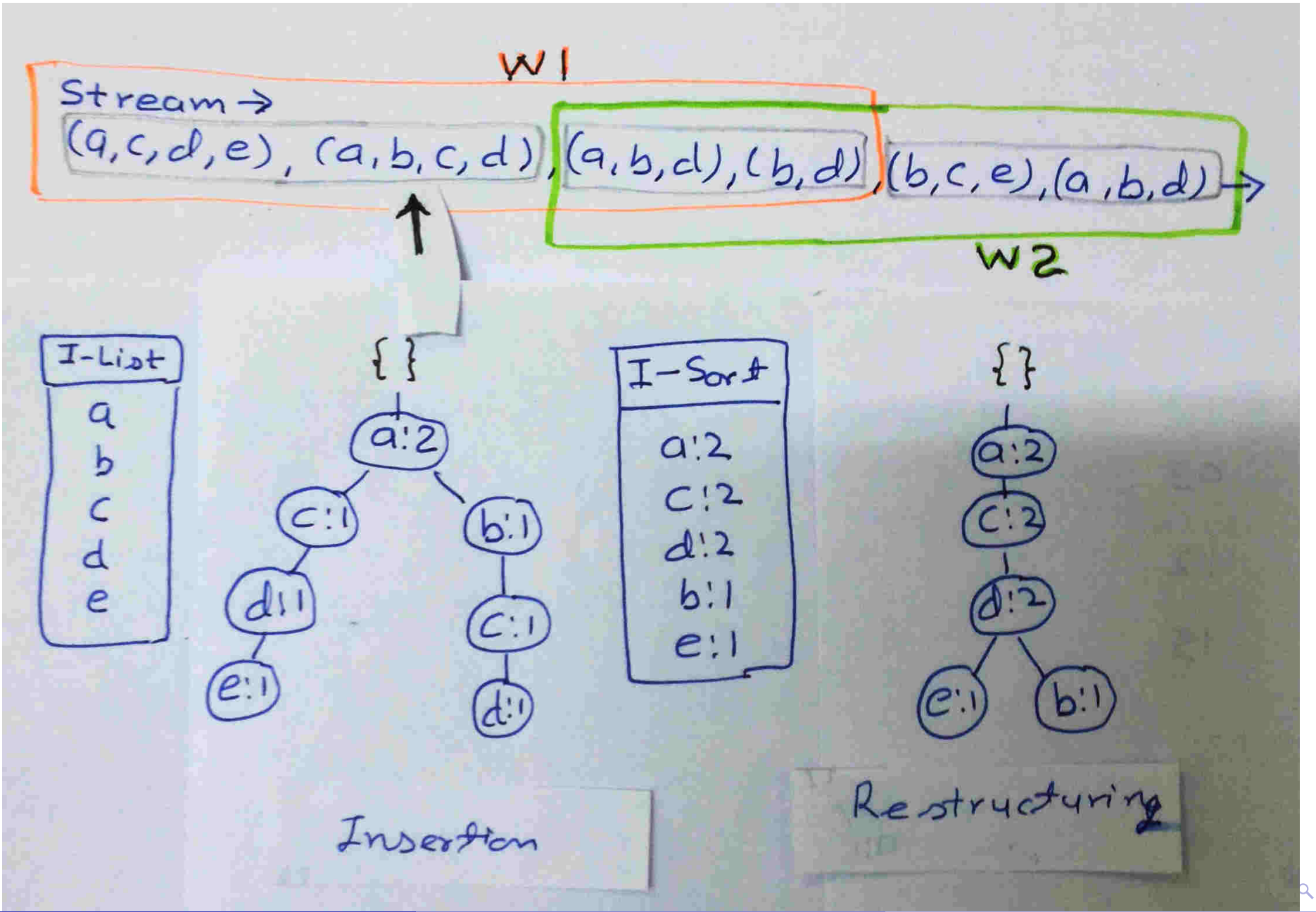
CPS-tree construction



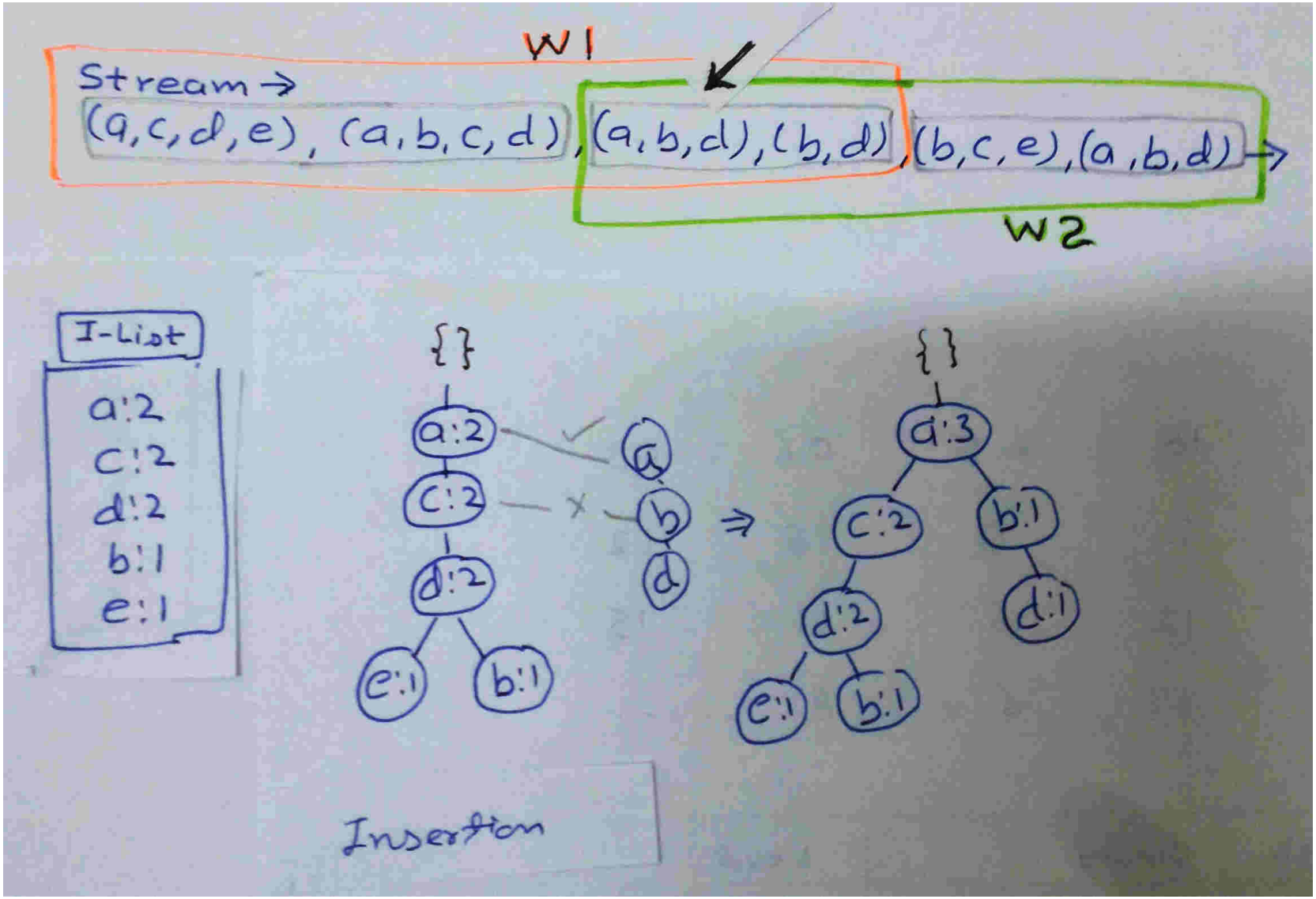
CPS-tree construction



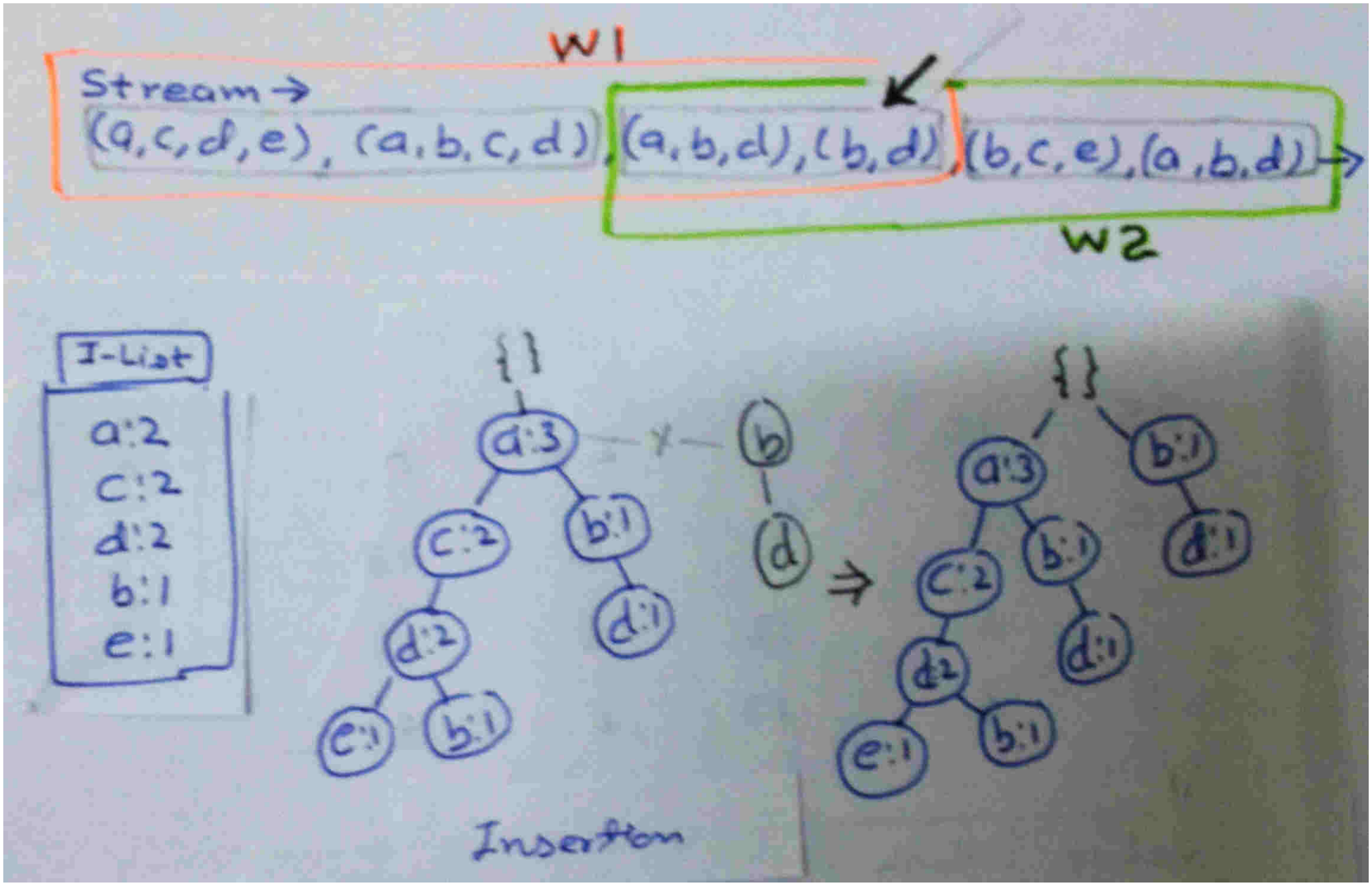
CPS-tree construction



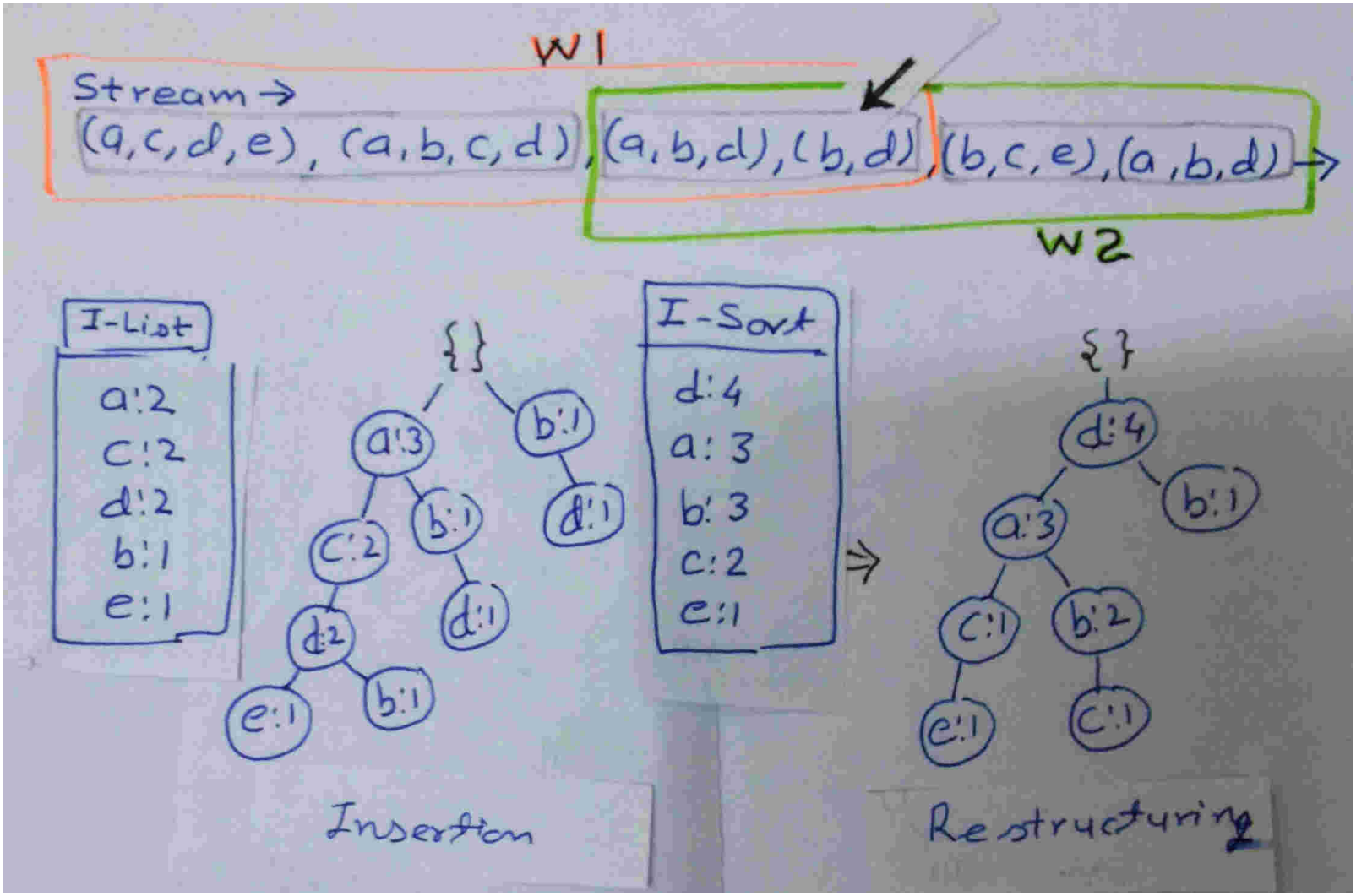
CPS-tree construction



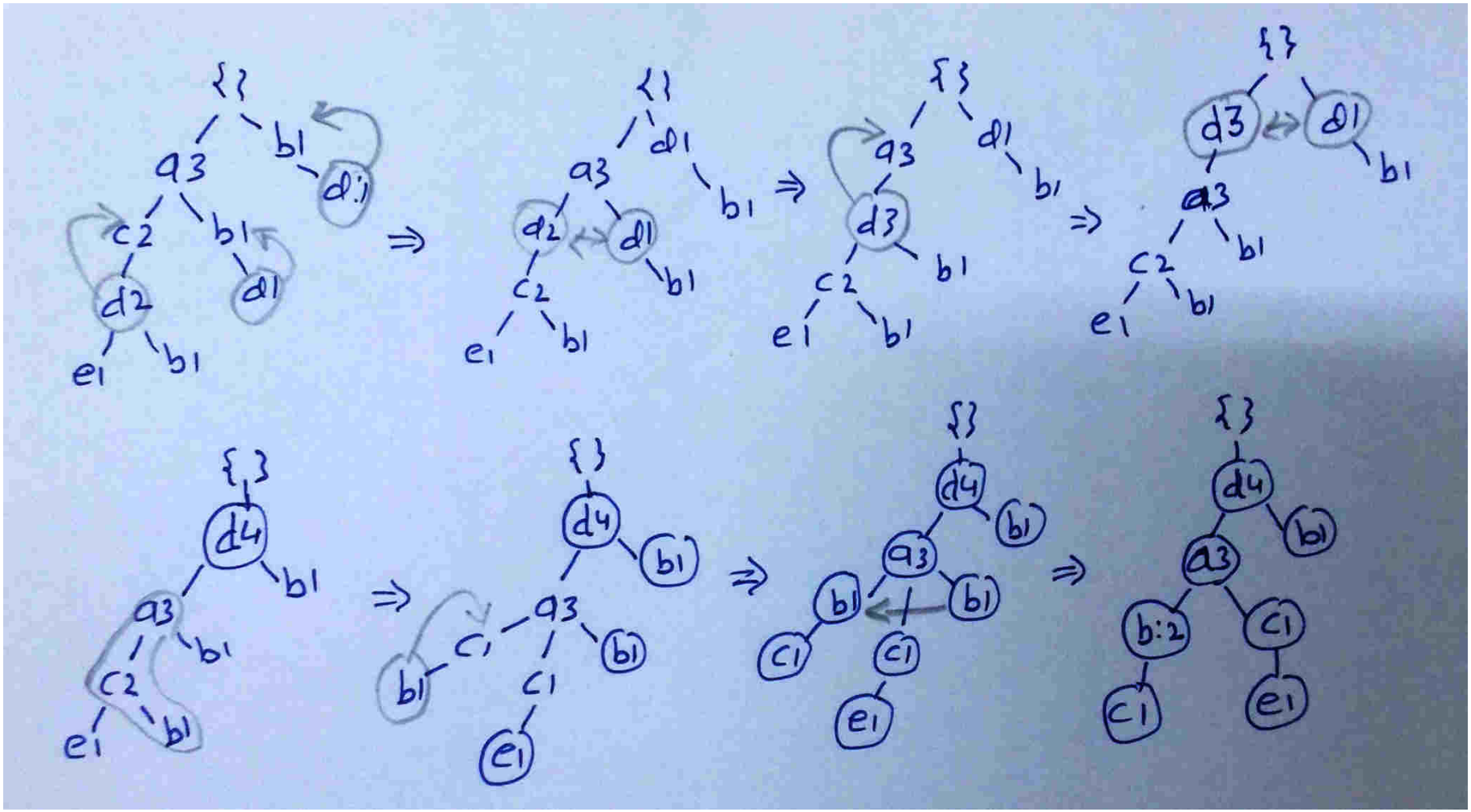
CPS-tree construction



CPS-tree construction

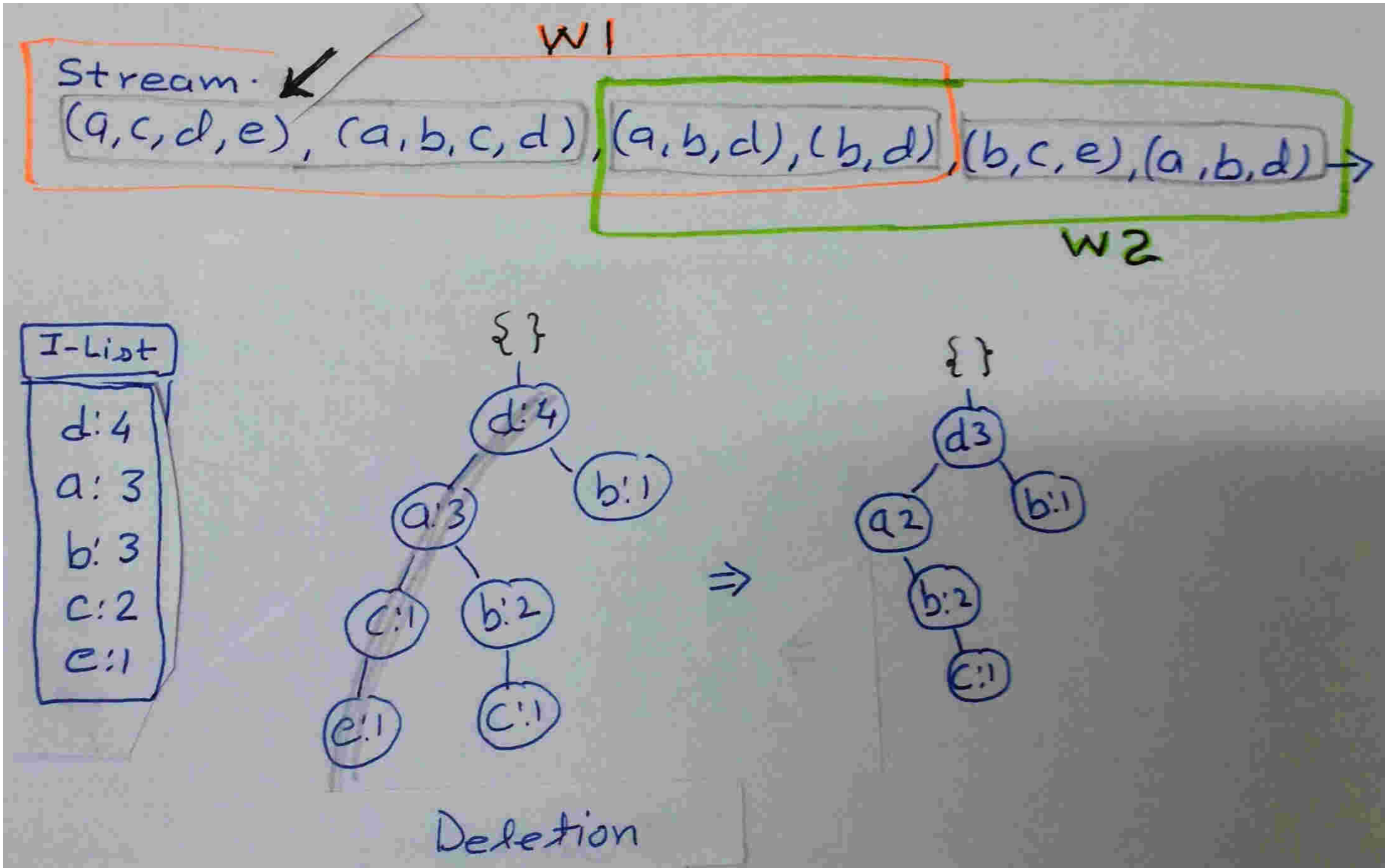


CPS-tree construction

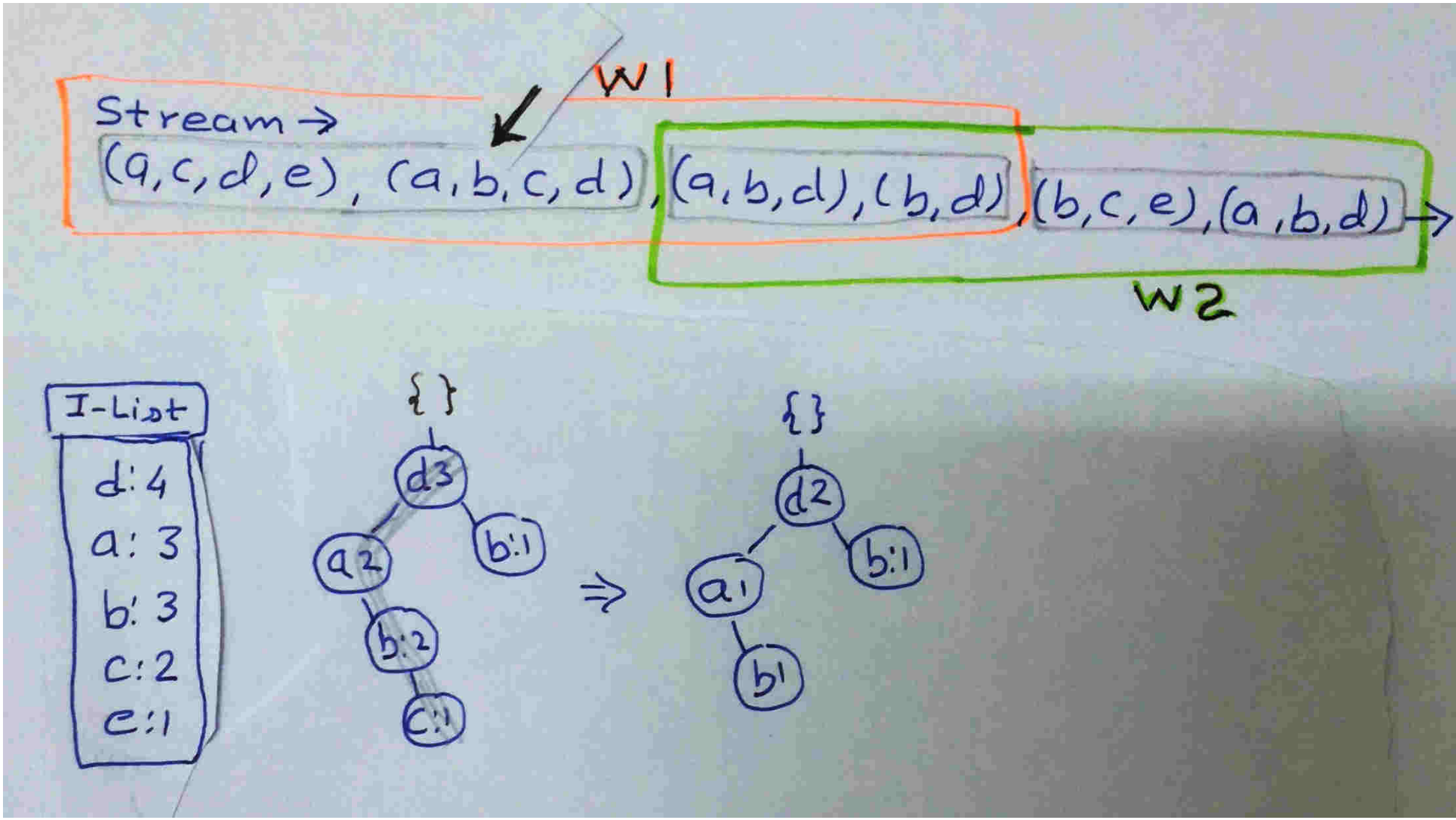


I-Sorted: d,a,b,c,e

CPS-tree construction

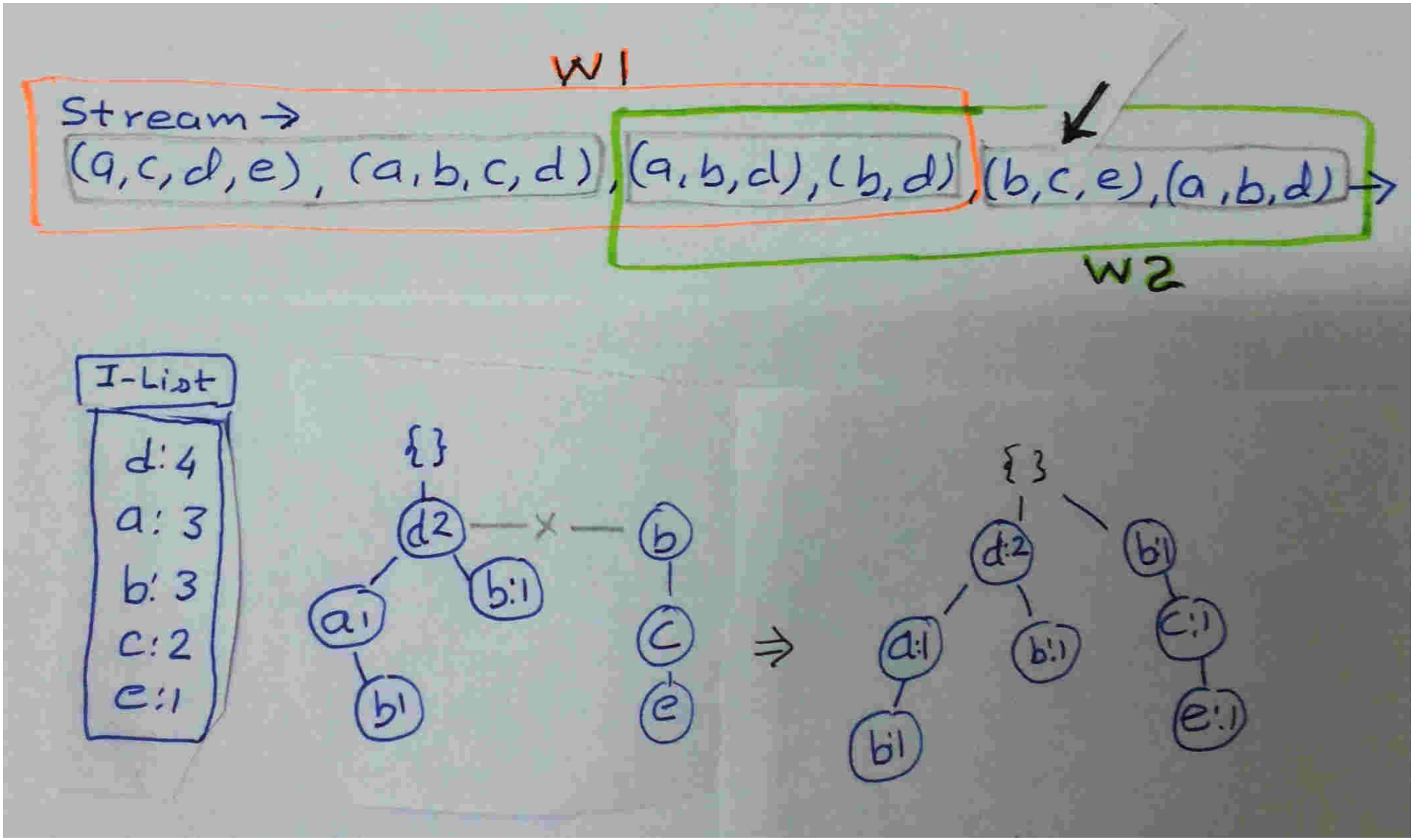


CPS-tree construction



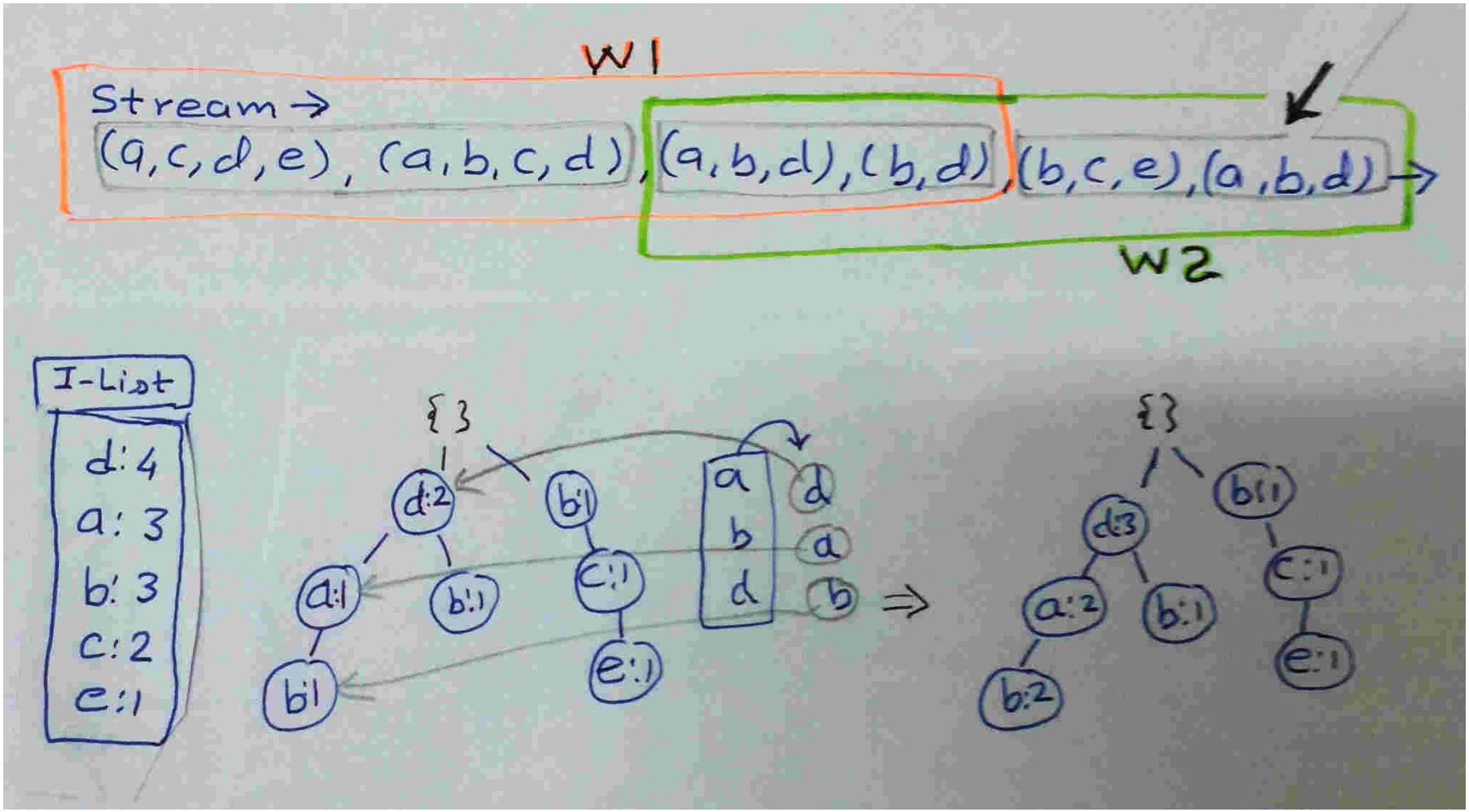
I-Sorted: d,a,b,c,e

CPS-tree construction



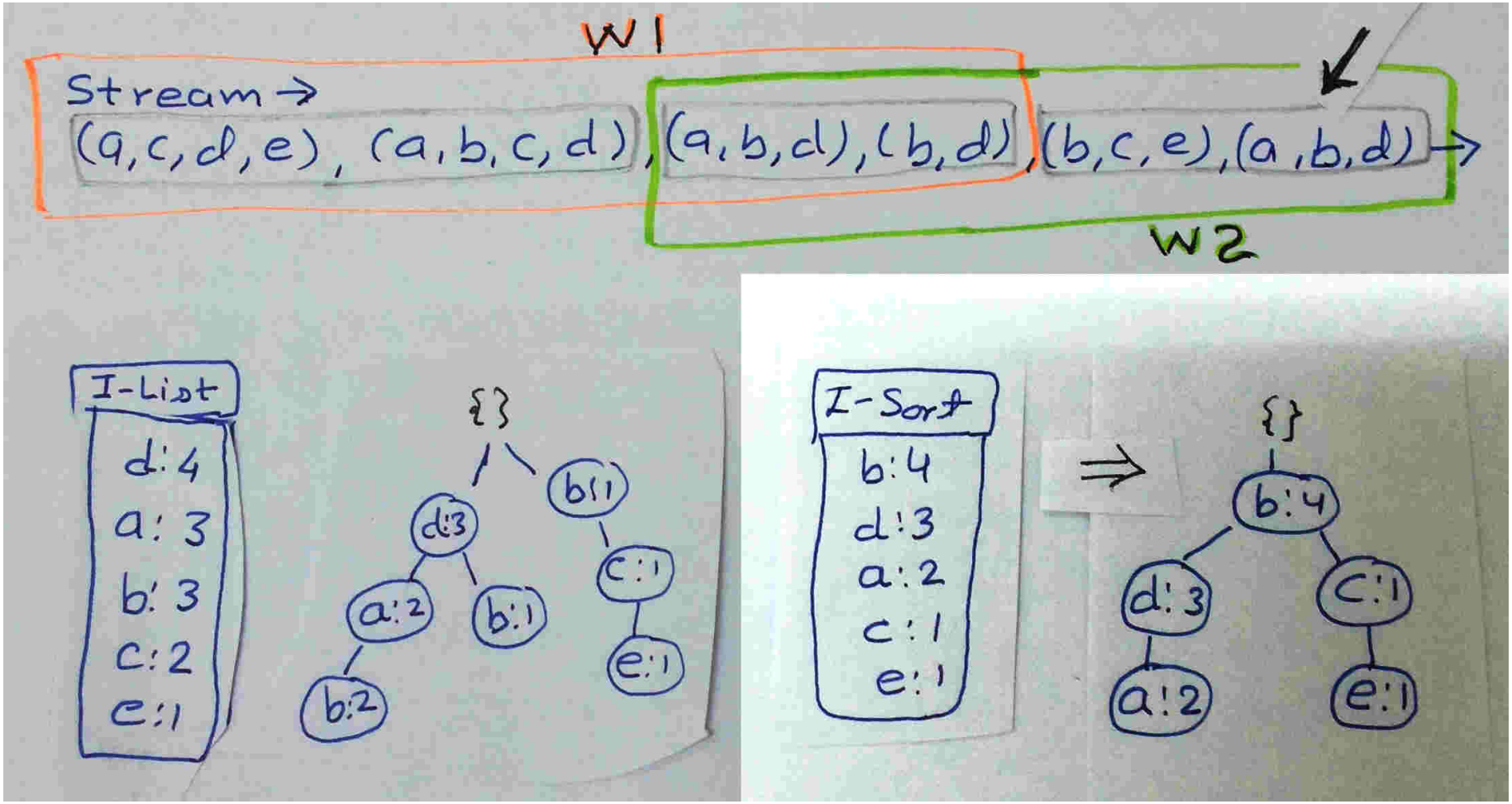
I-Sorted: d,a,b,c,e

CPS-tree construction



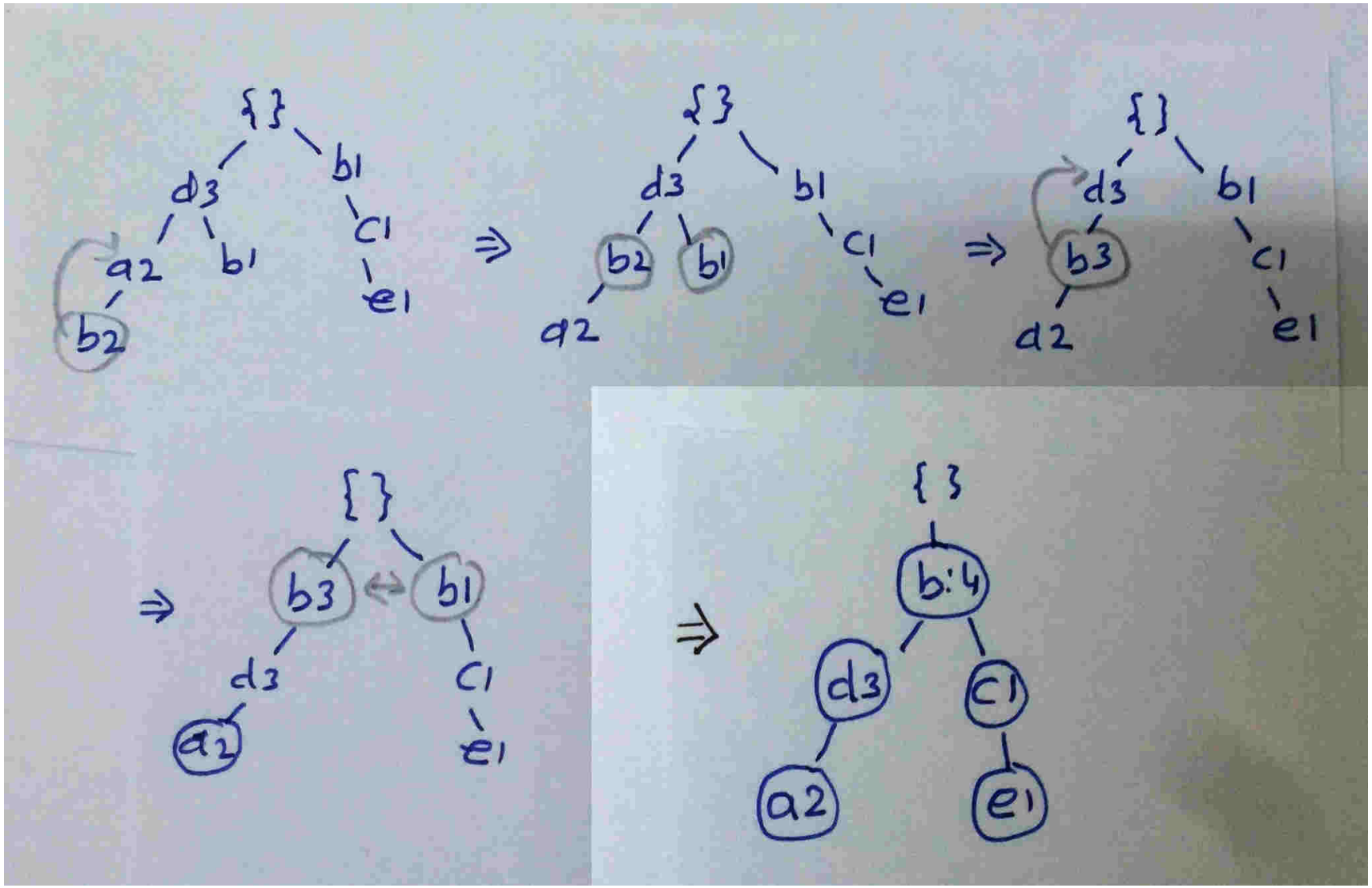
I-Sorted: d,a,b,c,e

CPS-tree construction



I-Sorted: b, d, a, c, e

CPS-tree construction



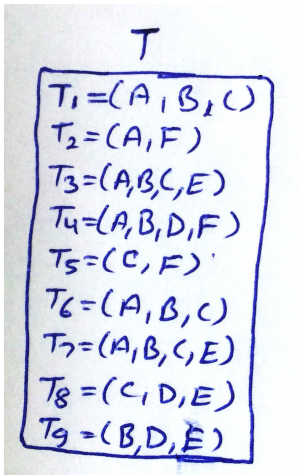
I-Sorted: b, d, a, c, e

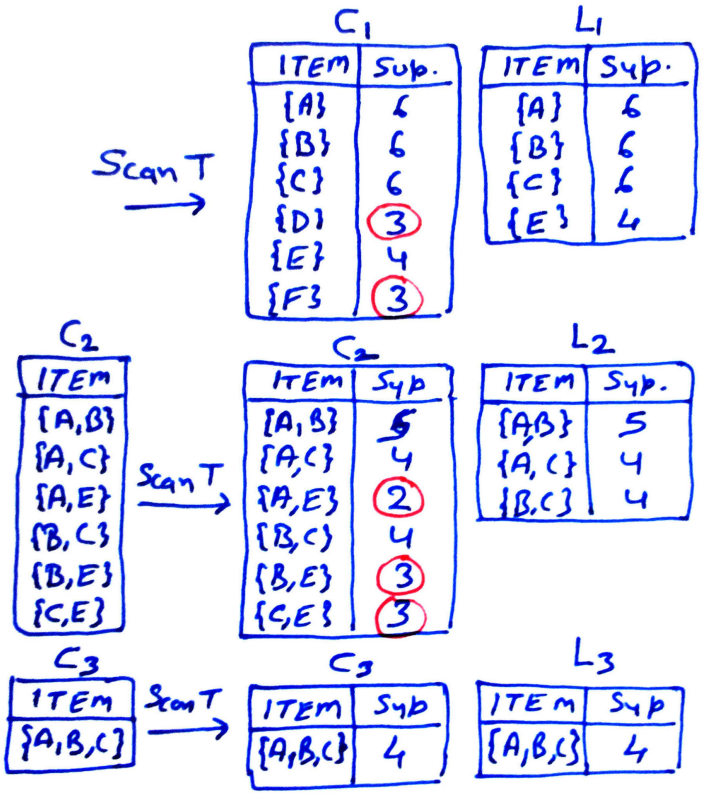
Data Mining

- Data mining is the process of extracting information. It is one of the part of KDD.
  (KDD involves collection of data, preprocessing, transformation, data mining, and interpretation)
  - **Predictive:** if we focus on new data (involves classification, regression, time series analysis, and prediction)
  - **Descriptive:** if we want to understand the data itself (involves clustering, summarization, association rules, or sequence discovery)
- Differs from traditional query processing
  - *Query* may not be well formed, even the miner may not be sure what he wants
  - *Data* may be preprocessed and modified so is in a different version
  - *Output* could be an analysis that may not be a subset of data
- **Issues:** Human interaction, Overfitting, Outliers, Large dataset, High dimension, Multimedia data, missing data, irrelevant data, noisy data, changing data.

Association Rule Mining

Given items $I = \{i_1, i_2, ..., i_m\}$ and set of transactions $T = \{t_1, t_2, ..., t_n\}$ discover **Association Rules** for minimum support.





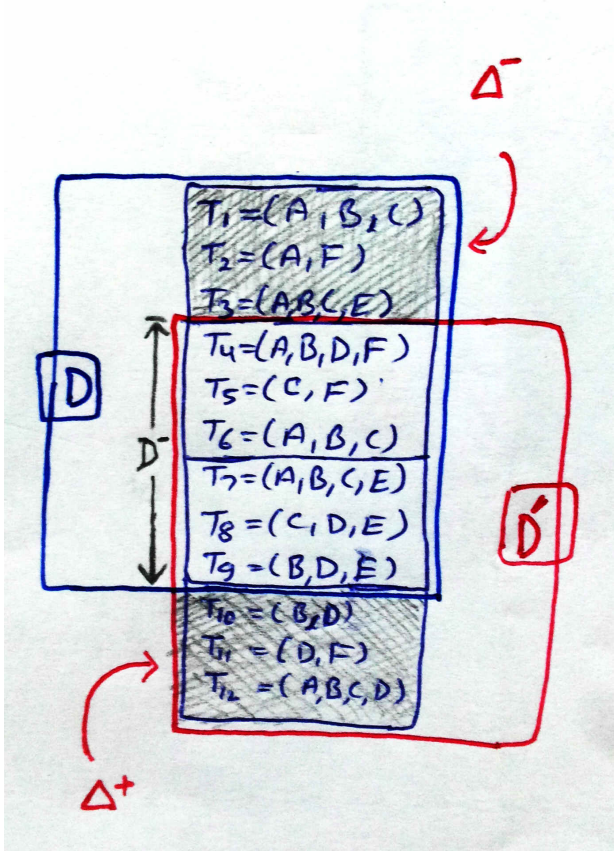
We have seen 1) **Apriori**, 2) Hash Based Algorithm DHP, and 3) Partition Based algorithms

Incremental AR Mining

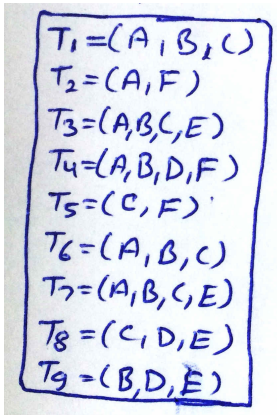
Incremental AR Mining: $\triangle^-$ portion of database D becomes obsolete and $\triangle^+$ more transactions are added.

Solution:

- **F**ast **UP**date algorithm (FUP)
- FUP2



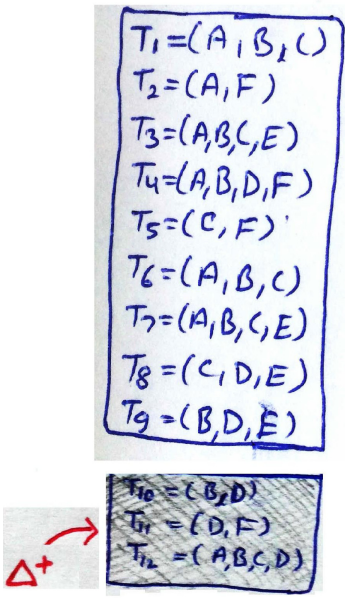
FUP



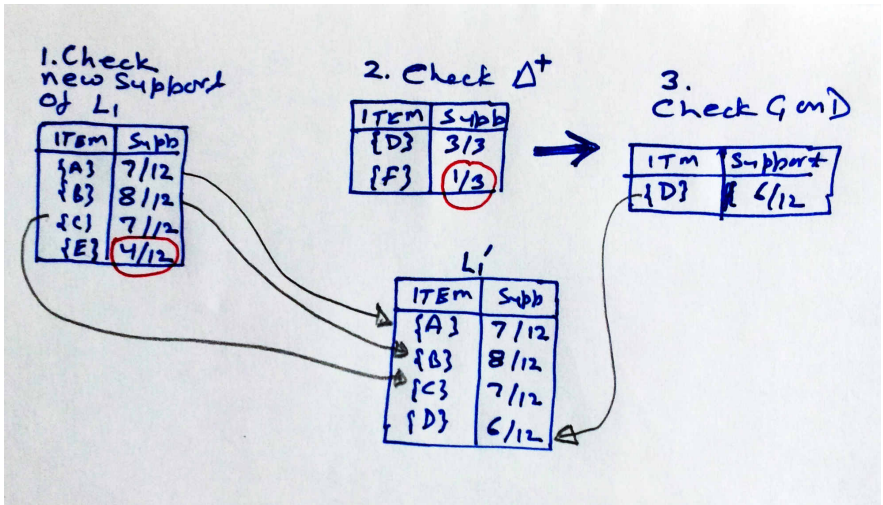
We have seen that frequent data set comes out to be

Item set	Support
{A}	6/9
{B}	6/9
{C}	6/9
{E}	4/9
{A B}	5/9
{A C}	4/9
{B C}	4/9
{A B C}	4/9

On arrival of $\triangle^+$ more transactions



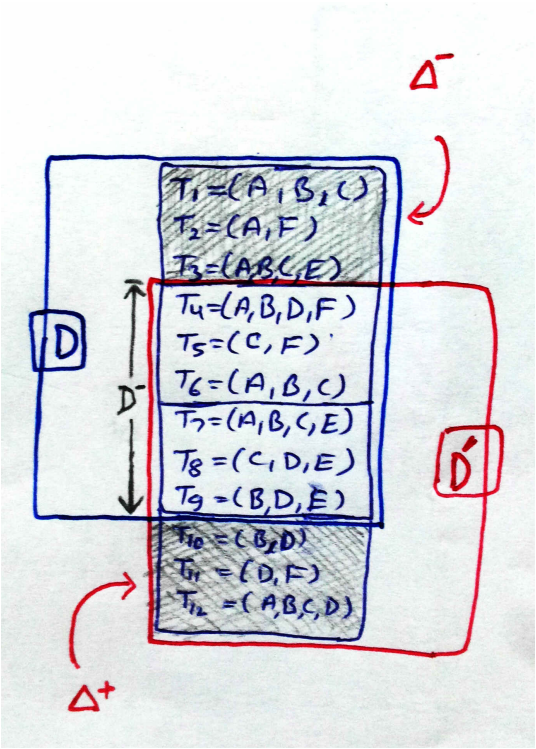
The first iteration, is as below.



FUP_2

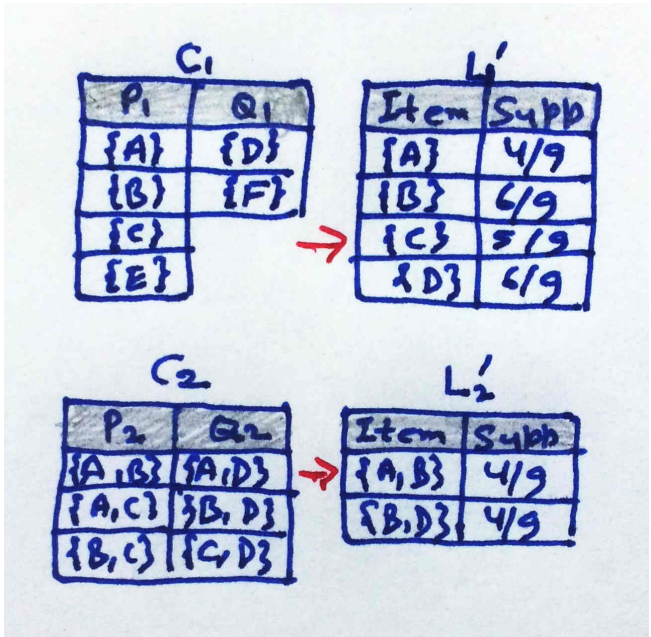
- FUP_2 can work for both $\triangle^-$ and $\triangle^+$
- L_k from previous mining result is used for dividing candidate itemset C_k into two parts
  - $P_k = C_k \cap L_k$
  - $Q_k = C_k - P_k$
- Itemset that is frequent in $\triangle^-$, must be infrequent in D^-.
- Further if itemset in Q_k in infrequent in $\triangle^+$ then it is infrequent in D^-.
- This technique helps to effectively reduce number of candidate itemsets.

FUP_2 at work



Item set	Support
{A}	6/9
{B}	6/9
{C}	6/9
{E}	4/9
{A B}	5/9
{A C}	4/9
{B C}	4/9
{A B C}	4/9

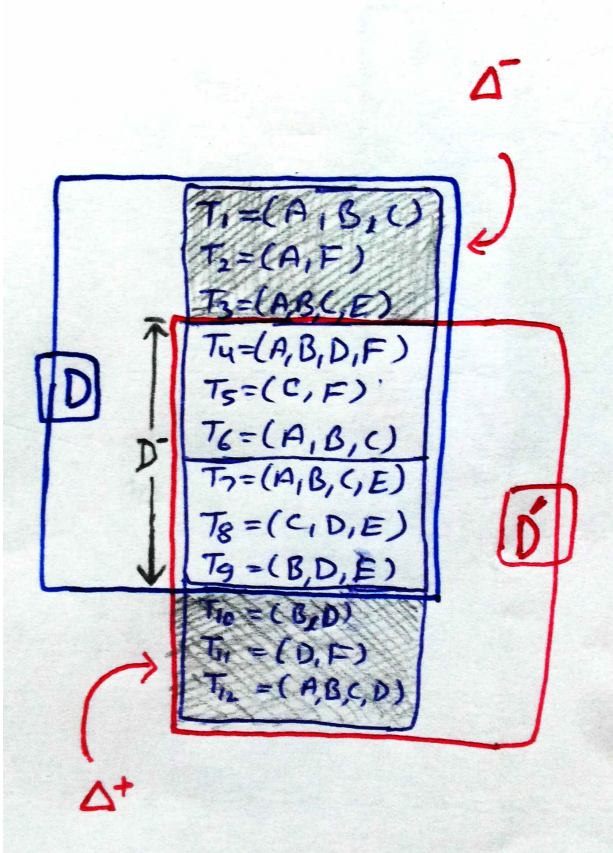
Frequent itemsets of D



- C_1 is set of all items. It is divided in P_i and Q_i
- Being frequent, support for all items in P_i is known. It could be updated using $\triangle^-$ and $\triangle^+$ only.
- $\text{Count}(\{A\})_{D'} = \text{Count}(\{A\})_D - \text{Count}(\{A\})_{\triangle^-} + \text{Count}(\{A\})_{\triangle^+}$
  $= 6 - 3 + 1 = 4$

FUP_2 at work

- In some cases only the scan of $\triangle^-$ and $\triangle^+$ is required.
- For example, $\text{Count}(\{F\})_{\triangle^+} - \text{Count}(\{F\})_{\triangle^-} = 0$ showing that support of $\{F\}$ can not be improved.
- Consequently, fewer itemsets have to be further scanned
- An iteration finishes when all the itemsets in P_i and Q_i are verified, and new set of frequent itemsets L'_i is generated



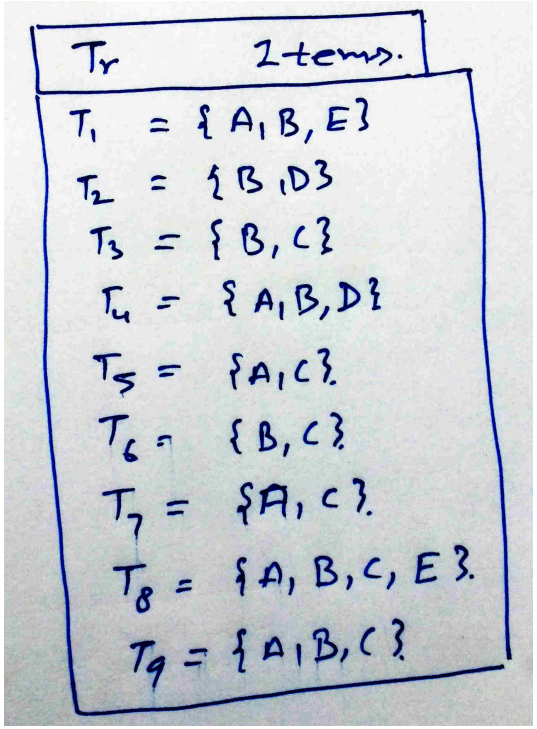
FUP_2H

Uses hashing for performance improvement

FP-tree

Frequent Pattern tree (FP-tree) structure[2] is an AR Mining technique

- Extended prefix tree structure
- Compression
- Avoid Candidate Generation
- Method is efficient and scalable



Let minimum support be 30%

Other methods are CATS-tree, CP-tree

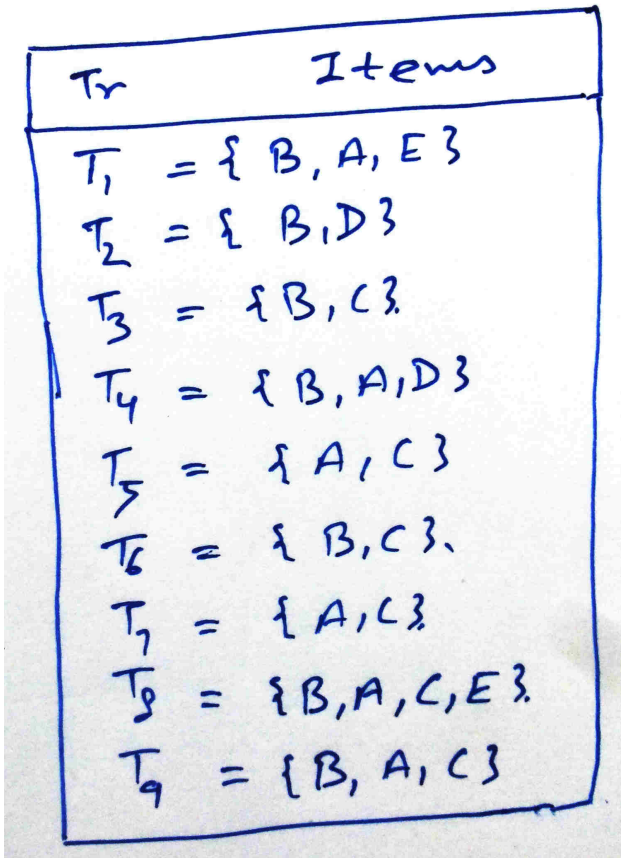
[2] Han, Jiawei and Pei, Jian and Yin, Yiwen, "Mining Frequent Patterns without Candidate Generation", in ACM Sigmod 29(2) pages 1-12, ACM 2000

FP-tree at work

Item	Frequency	Priority
A	6	2
B	7	1
C	6	3
D	2	4
E	2	5

This would lead to a reordering

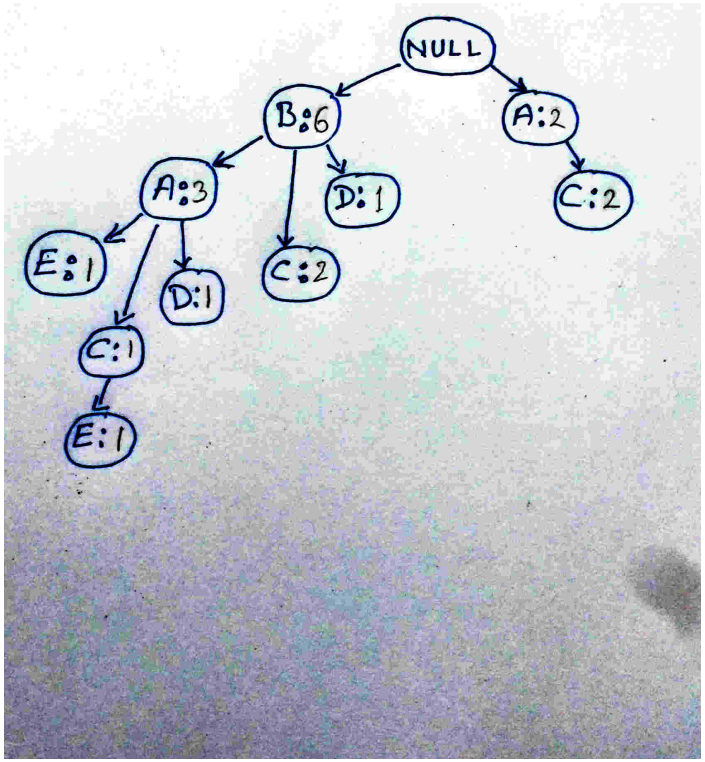
Priority order: B, A, C, D, E



FP-tree at work

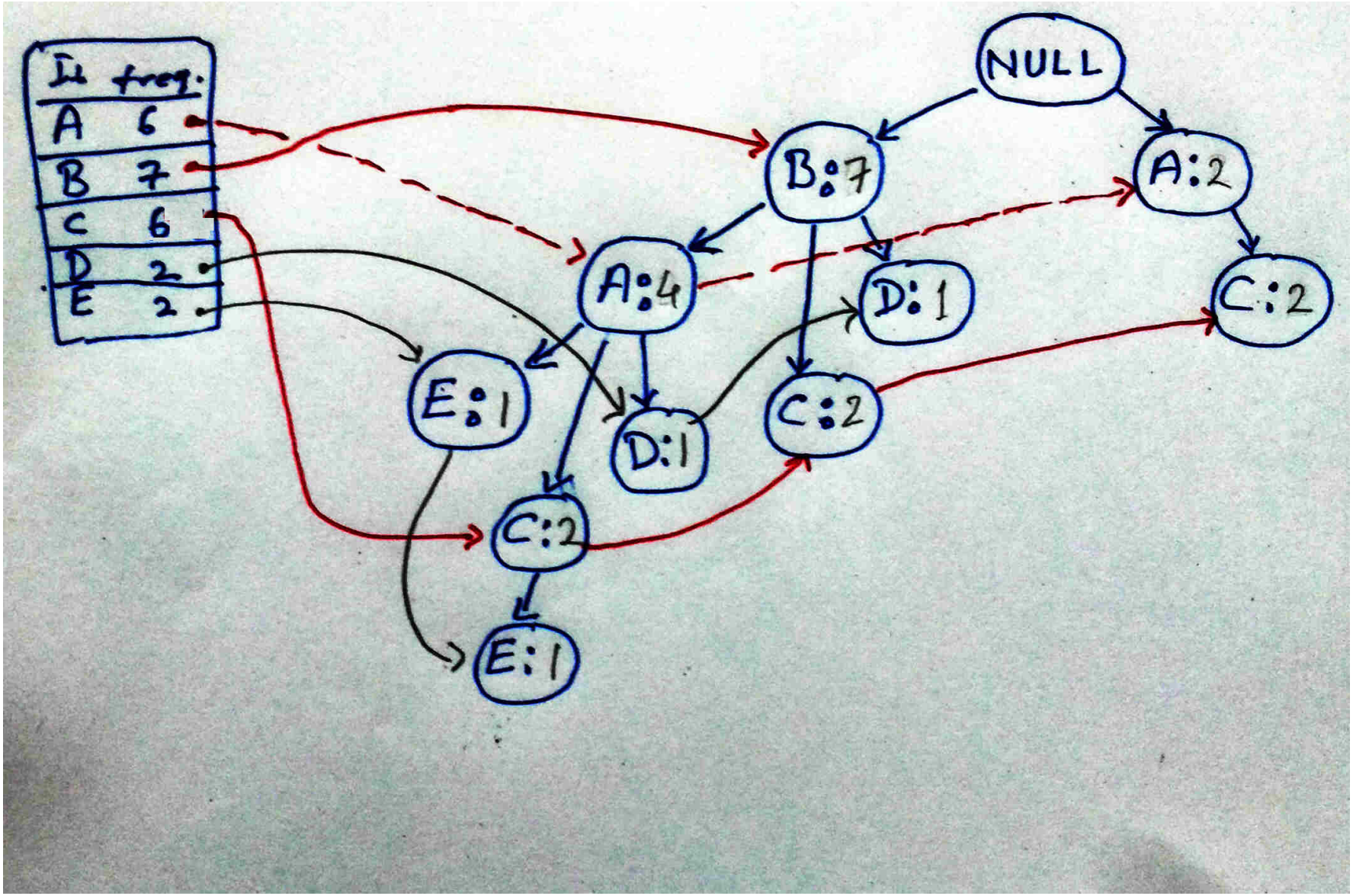
Tr	Items
T_1	= {B, A, E}
T_2	= {B, D}
T_3	= {B, C}
T_4	= {B, A, D}
T_5	= {A, C}
T_6	= {B, C}
T_7	= {A, C}
T_8	= {B, A, C, E}
T_9	= {B, A, C}

FP-tree with transaction {B,A,C,E}



Next transaction {B,A,C}

FP-tree at work



Create links

FP-tree at work

Item	Conditional Pattern Base	Cond. FP-Tree
E	{B,A:1}, {B,A,C:1}	(<B:2,A:2>)
D	{B,A:1}, {B:1}	(<B:2>)
C	{B,A:2}, {B:2}, {A:2}	(<B:4,A:2>, <A:2>)
B	{}	(Null)
A	{B:4}	(<B:2>)

- **E:** {B,E}, {A,E}, {A,B,E}
- **D:** {B,D}
- **C:** {B,A,C}, {B,C}, {A,C}
- **A:** {B,A}

CATS Tree

- CATS (Compressed and Arranged Transaction Sequences Tree) Tree [3] extends the idea of FP-Tree to improve storage compression and allow frequent pattern mining without generation of candidate itemsets.
- Frequent pattern mining with different supports need not to rebuilding
- Works in a single pass over the database and can handle insertion/deletion

CATS Tree	**FP-Tree**
Contains all items in every transaction	Contains only frequent items
Sub-trees are locally optimized to improve compression	Sub-trees are not locally
Ordering of items within paths from the root to leaves are ordered by local support	Ordering of items within paths from the root to leaves are ordered by global support
CATS nodes of the same parent are sorted in descending order according to local frequencies	Children of a node are not sorted

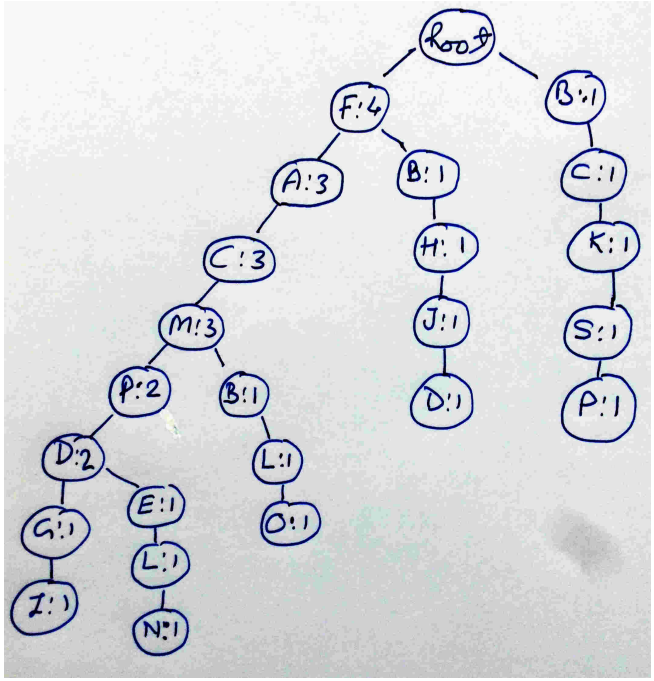
[3] Cheung, William and Zaiane, Osmar R, "Incremental mining of frequent patterns without candidate generation or support constraint", in Seventh International Database Engineering and Applications Symposium, pages 111–116, IEEE, 2003

CATS Tree

Allow frequent pattern mining without generation of candidate itemsets, in single pass and can handle insertion/deletion.

Consider database as

TID	Transaction
T1	F,A,C,D,G,I,M,P
T2	A, B, C, F, L, M, O
T3	B, F, H, J, O
T4	B, C, K, S, P
T5	A, F, C, E, L, P, M, N



Issue is efficiency and compactness

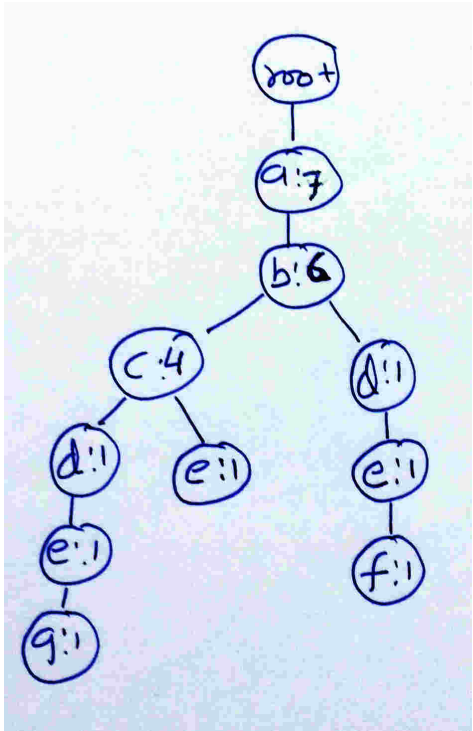
CanTree

Nodes of **Canonical-order Tree** follow canonical order. It is easy to search patterns and is more compact.

Consider transaction database as

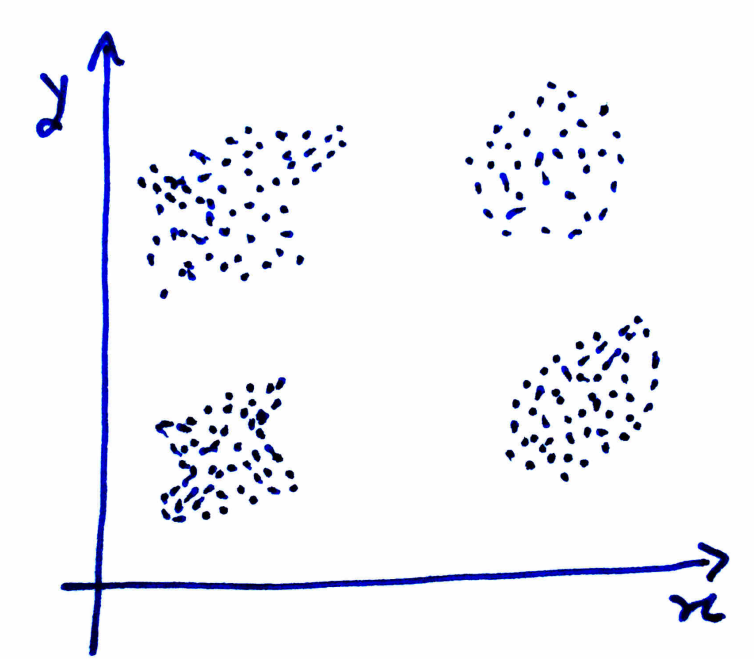
ID	Transaction
T1	a, d, b, g, e, c
T2	d, f, b, a, e
T3	a
T4	d, a, b
T5	a, c, b
T6	c, b, a, e
T7	a, b, c

←



Clustering in a dynamic database

- Can we use *k*-Means clustering algorithm?



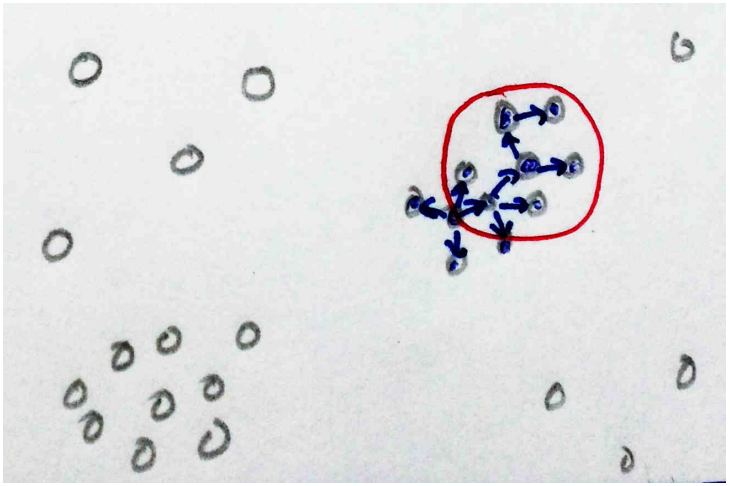
  - Randomly choose *k* data points as centroid
  - Assign each data point to closest centroid
  - Update centroid until converge
  - Typically SSD (sum of square error) is optimized

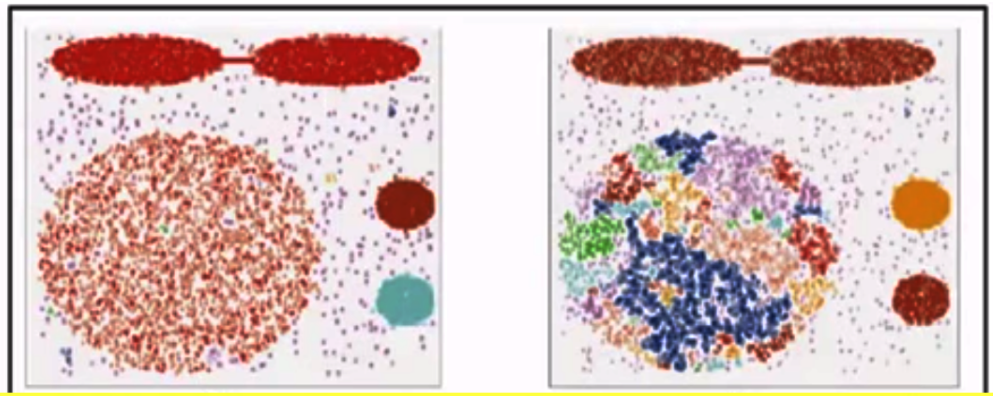
- What about PAM (partitioning around medoids)? NO
- Single/Average/Farthest link clustering?

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a spatial clustering algorithm of KDD96

- Parameters (Eps/MinPts) and points (core/border/noise)
- Uses DFS

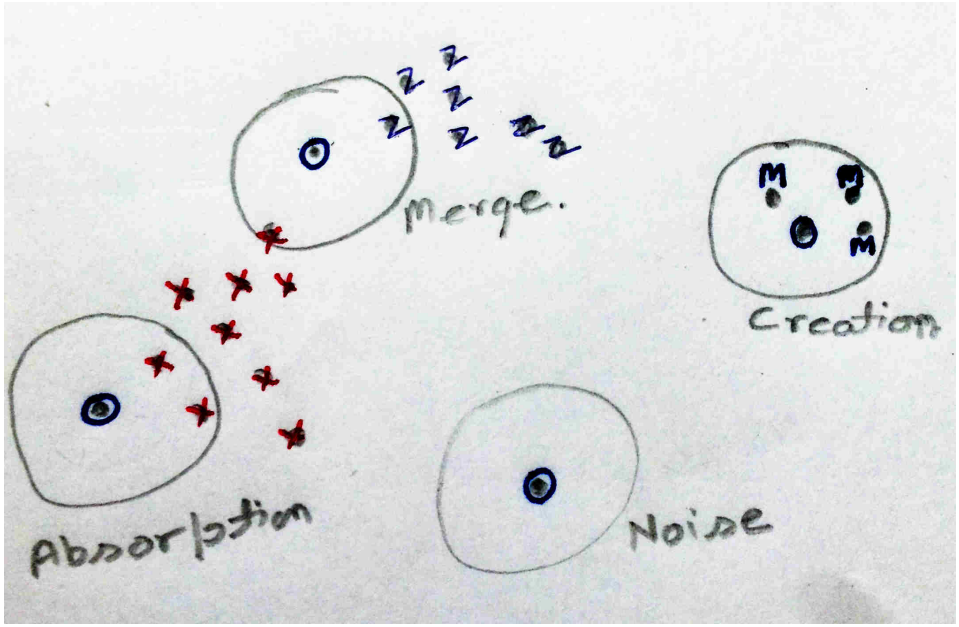




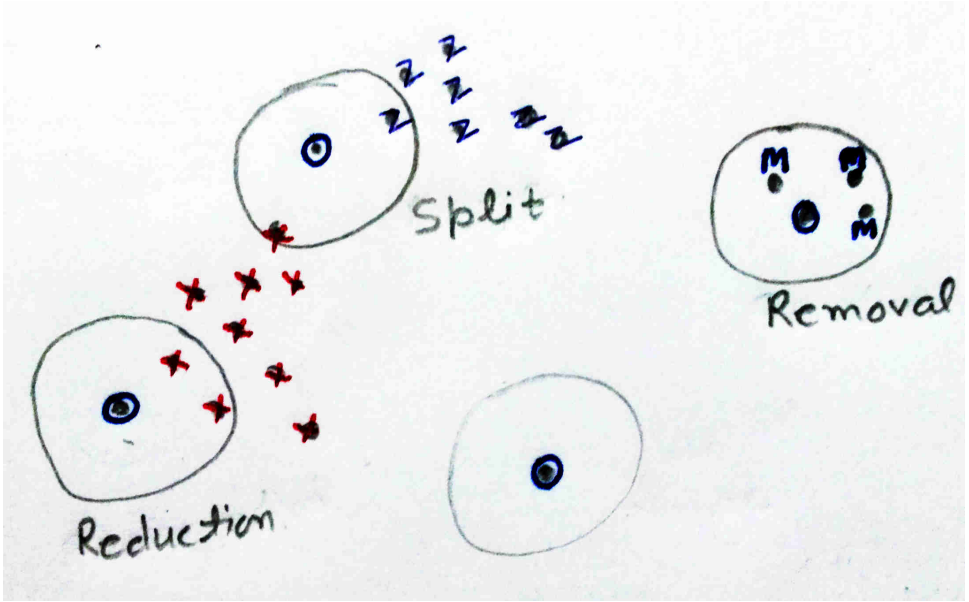
Figures from G. Karypis, E.-H. Han, and V. Kumar, *COMPUTER*, 32(8), 1999

- Advantage: clusters of arbitrary shape
- Disadvantage: Sensitive to parameters

Incremental DBSCAN



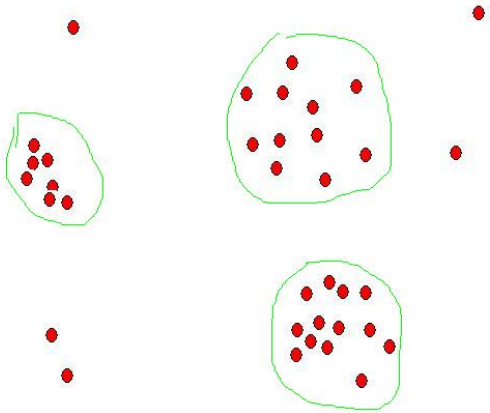
Data Arrival



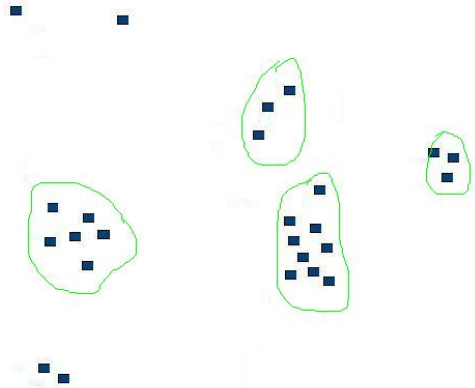
Date Departure

- Parameters: Eps, MinPts
- Data Points: core, border or noise

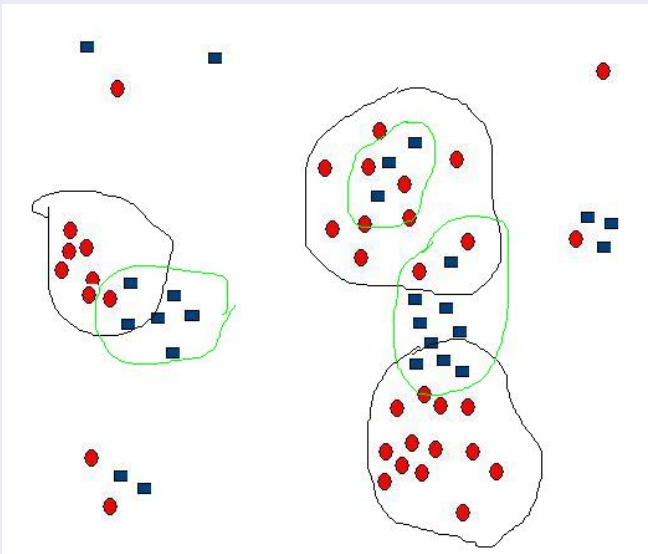
Incremental DBSCAN

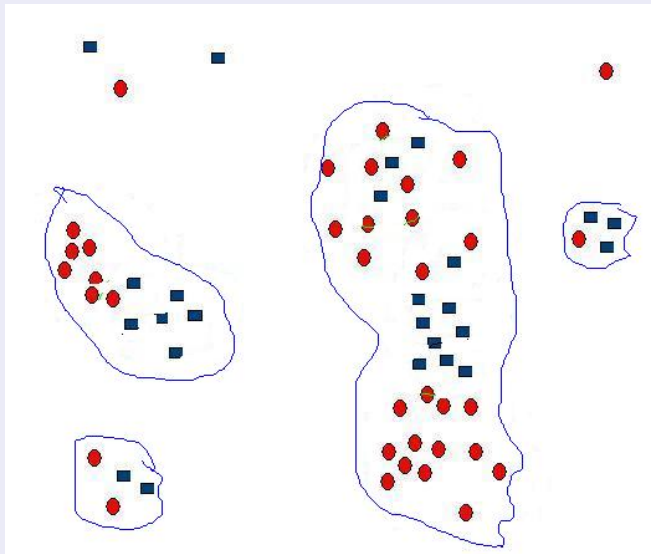


Initial Database



New Date

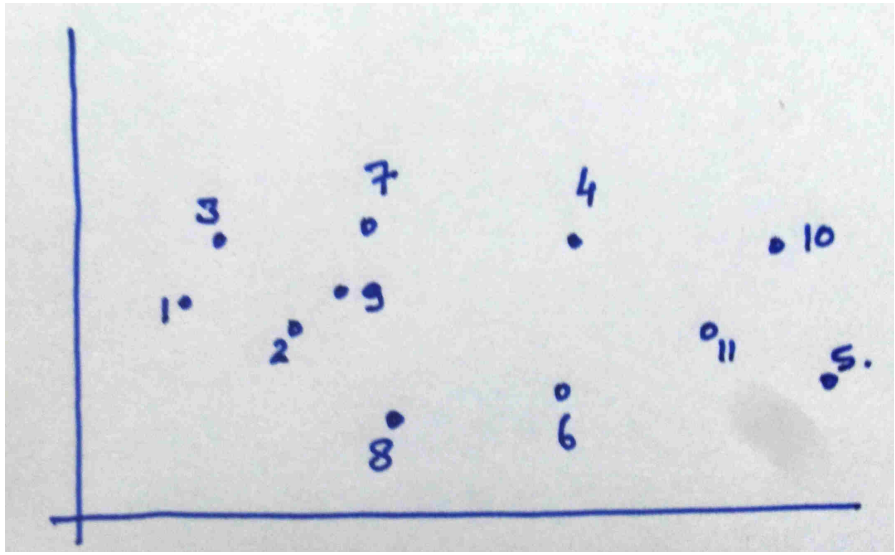


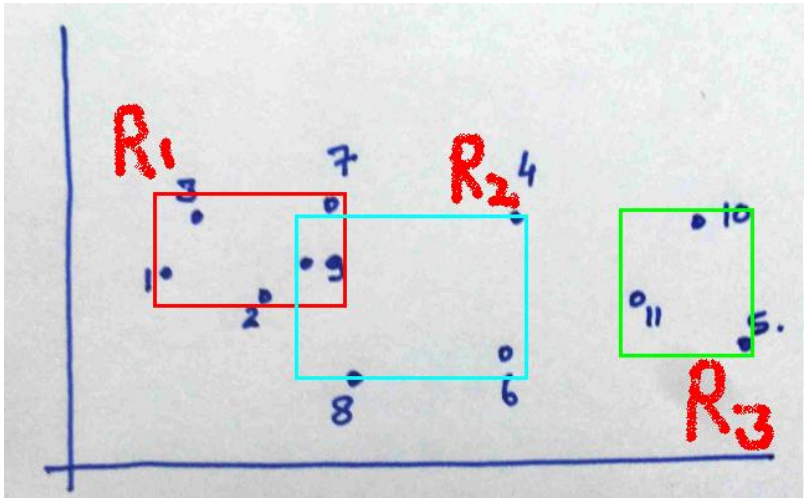


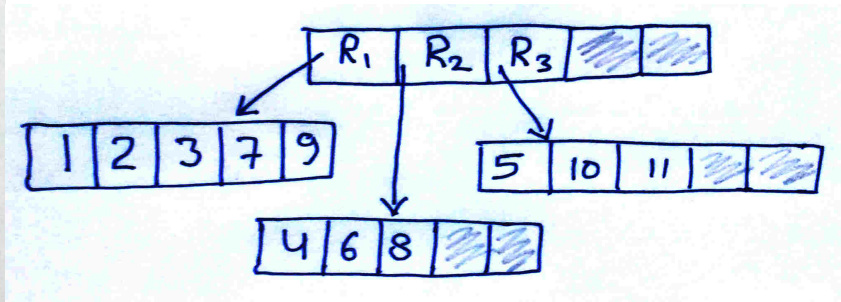
R-tree date structure

Consider arrival of

- P_1, P_2, P_3, P_4, P_5
- P_6 (split and region formation)
- P_7 (R1 expands)
- P_8 (R2 expands)
- P_9, P_{10}
- P_{11} (Split in R2)







Data streams

Stream has data in rapid succession and no re-scan is possible. Even storage space is insufficient to accommodate all datum.

Without storing all the data one wish to estimate

- Set of frequent items
- Number of distinct items
- Frequent itemsets
- *etc*

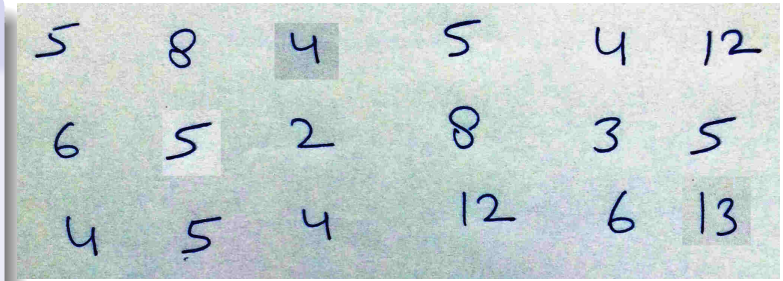
Frequent items over data stream

Assume general arrival model of data stream of size $m = \sum_i f_i$

- Item i of frequency f_i is frequent, if $f_i > m/(k+1)$ for some k
- At most k frequent items are possible
- Lower bound of space is $\log(\binom{n}{k})$ bits

Insert x in data structure

1. if (A.ismember(x)) A[x]++
2. else A.insert(x)
3. if(A.size == k+1) then $\forall y \in A$
4. $A[y] - -$,
5. if (A[y] ==0) A.delete(y);



Index	Item	Frequency
1		
2		
3		
4		
5		

When stream is fast;

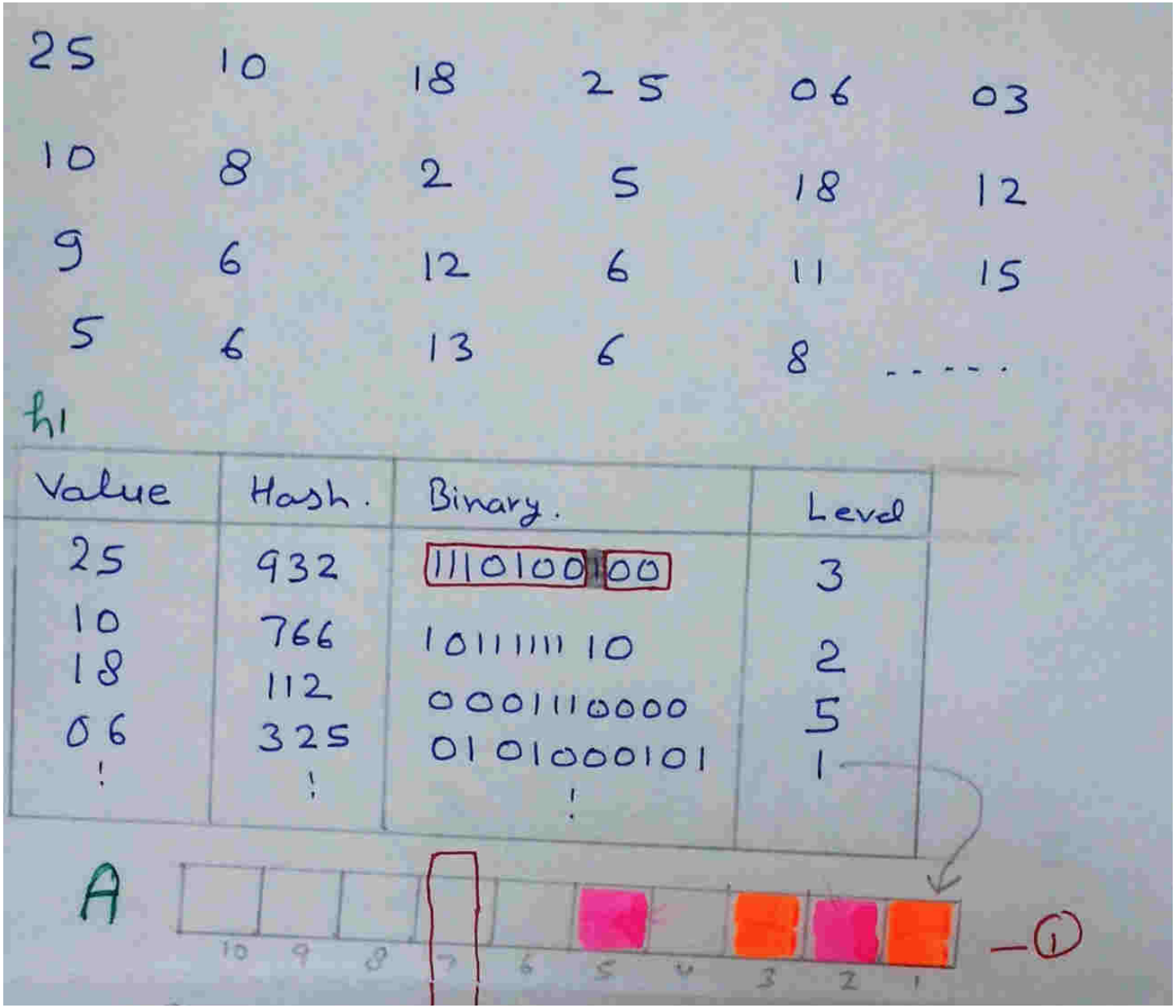
Do Sampling: $\hat{f}_i = q_i/p$, where q_i is sampling estimate.

Count distinct over data streams

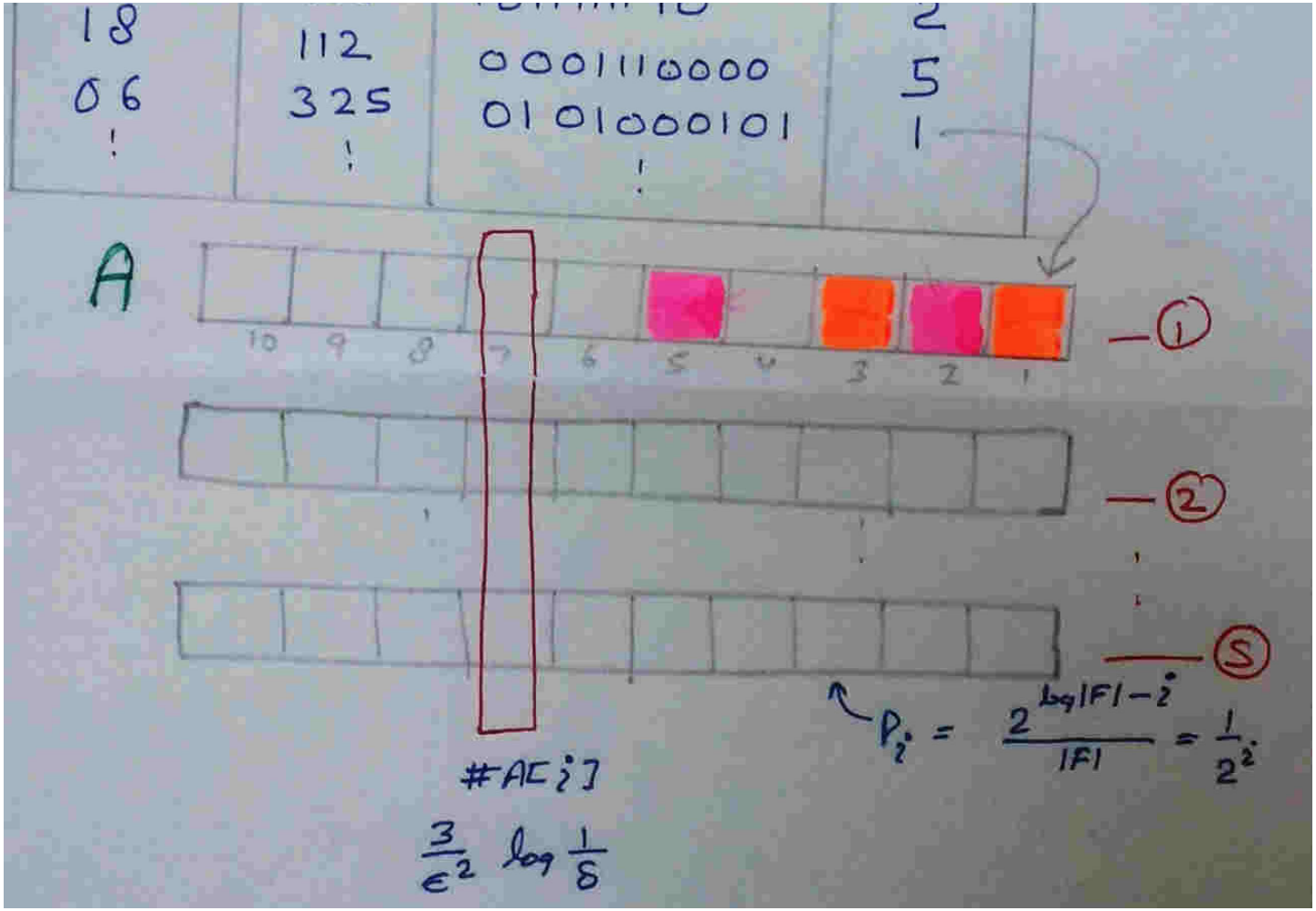
- If $x = ???..??1\overbrace{000...0}^{i-1}$ then L[x]=i
- Probability of L[x]=i is $p_i = \frac{2^{\log|F|-i}}{|F|} = 1/2^i$ when $x \in \{1, 2, .., F\}$
- FM sketch, a bitmap A of size $\log|F|$ with hash function h is used
- Arrival of x sets bit $A[L[h(x)]] = 1$. Probability of $A[i] = 1$ after seeing n items is $1 - (1 - p_i)^n$
- With s independent copies of FM sketch, let $\#A[i]$ represent count of 1's at level i and $\hat{q}_i = \#A[i]]/s$. Then choose i, such that $\hat{q}_I \geq \frac{3}{\epsilon^2}\log\frac{1}{\delta}$. By Chernoff's bound $x \in (1+\epsilon)E[x]$ with probability $(1-\delta)$

$$\hat{n} = \frac{\log(1-\hat{q}_i)}{\log(1-p_i)}$$

Count distinct



Count distinct over data streams



$$\hat{n} = \frac{\log(1 - \hat{q}_i)}{\log(1 - p_i)}$$

Frequent pattern mining over data streams

- Applications involves retail market data analysis, network monitoring, web usage mining, and stock market prediction.
- Using sliding window
- Efficiently remove the obsolete, old stream data
- Compact Pattern Stream tree (CPS-tree) [4]
- Highly compact frequency-descending tree structure at runtime
- Efficient in terms of memory and time complexity
- Pane and window
- Insertion and restructuring

[4] Tanbeer, Syed Khairuzzaman and Ahmed, Chowdhury Farhan and Jeong, Byeong-Soo and Lee, Young-Koo, "Sliding window-based frequent pattern mining over data streams", in Information sciences, 179(22) pages 3843–3865, Elsevier, 2009

Thank You!

Thank you very much for your attention!

Queries ?