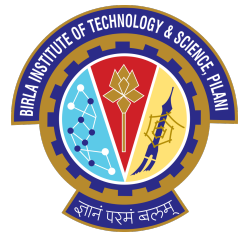
SS-ZG548: ADVANCED DATA MINING

Lecture-09: Stream Mining



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

Feb 17, 2018 (WILP @ BITS-Pilani Jan-Apr 2018)

Recap: Frequent items over data stream

- Wish to output a list of items such that $f_i > m/(k+1)$ where m is the size of stream and k frequent items are sought

Maintain a data structure A and update over stream as below

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);

- Memory requirement: of the order of k

Count distinct over data streams (FM sketch)

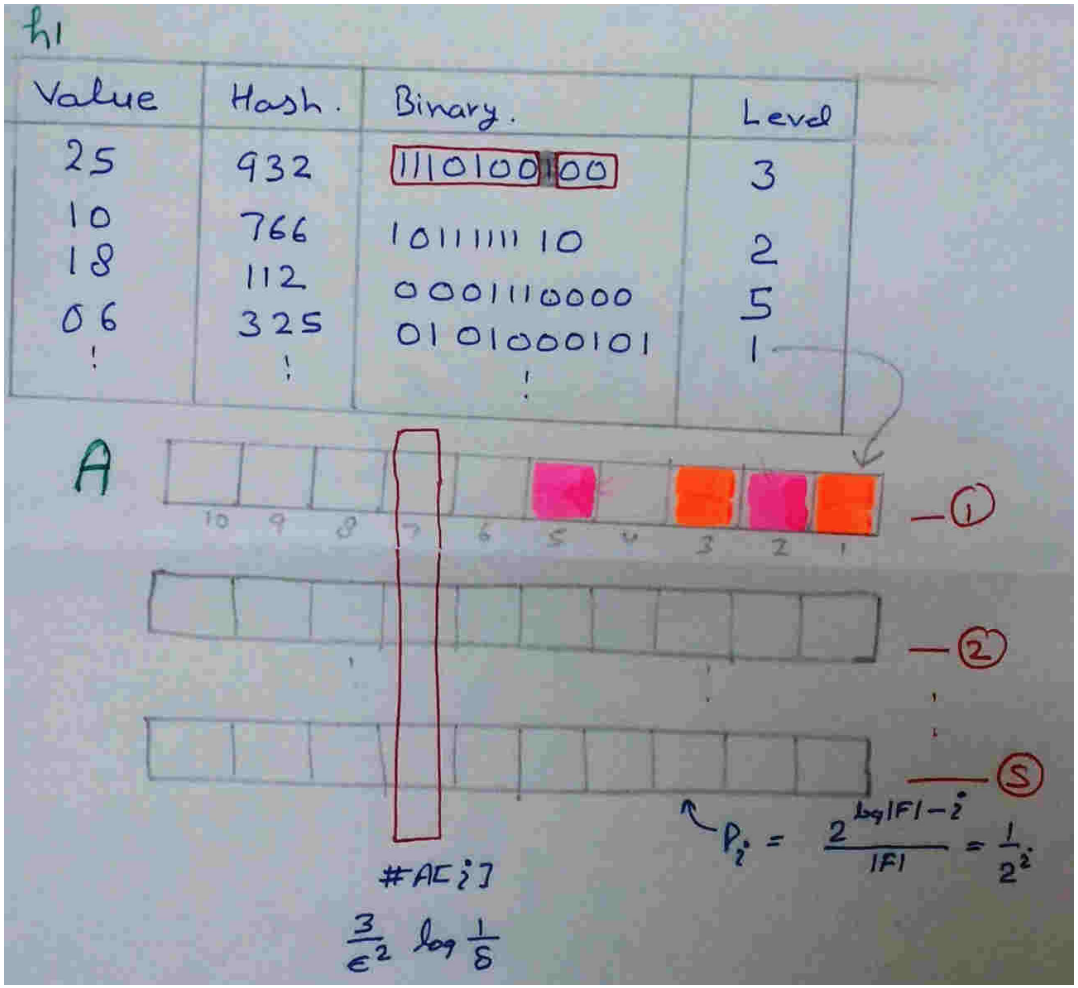
Estimate number of distinct items in data stream

- 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 is a bitmap A of size $\log|F|$ with hash a function h
- Arrival of an item x, sets bit $A[L[h(x)]] \leftarrow 1$.
  Probability that $A[i] = 1$ 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 $\hat{q}_i = \frac{\#A[i]}{s}$. Then choose i, such that $\hat{q}_i \geq \frac{3}{\epsilon^2}\log\frac{1}{\delta}$. By Chernoff's bound $x \leq (1+\epsilon)E[x]$ with probability $(1-\delta)$

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

In action: Count distinct over data streams

Consider a data stream: 25, 10 ,18 ,25, 06, 03, 10, 8, 2, 5, 18, 12, 9, 6, 12, 6, 11, 15, 5, 6, 13, 6, 8, ... $\rightarrow$



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

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) [1]
- Highly compact frequency-descending tree structure at runtime
- Efficient in terms of memory and time complexity
- Pane and window
- Insertion and 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

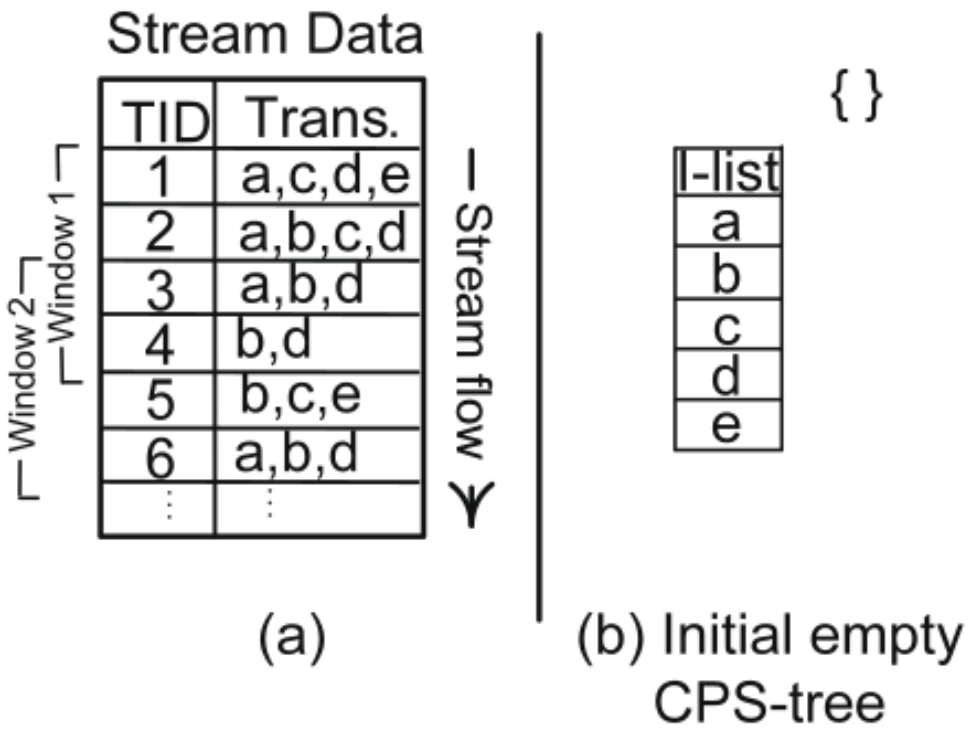
CPS-tree construction

Algorithm 1 *(Construction of a CPS-tree for a data stream)*
Input: *Stream_data, Pane_size, Window_size, Initial_Sort_Order*
Output: T_{sort}: *a CPS-tree for the current window*
Method:

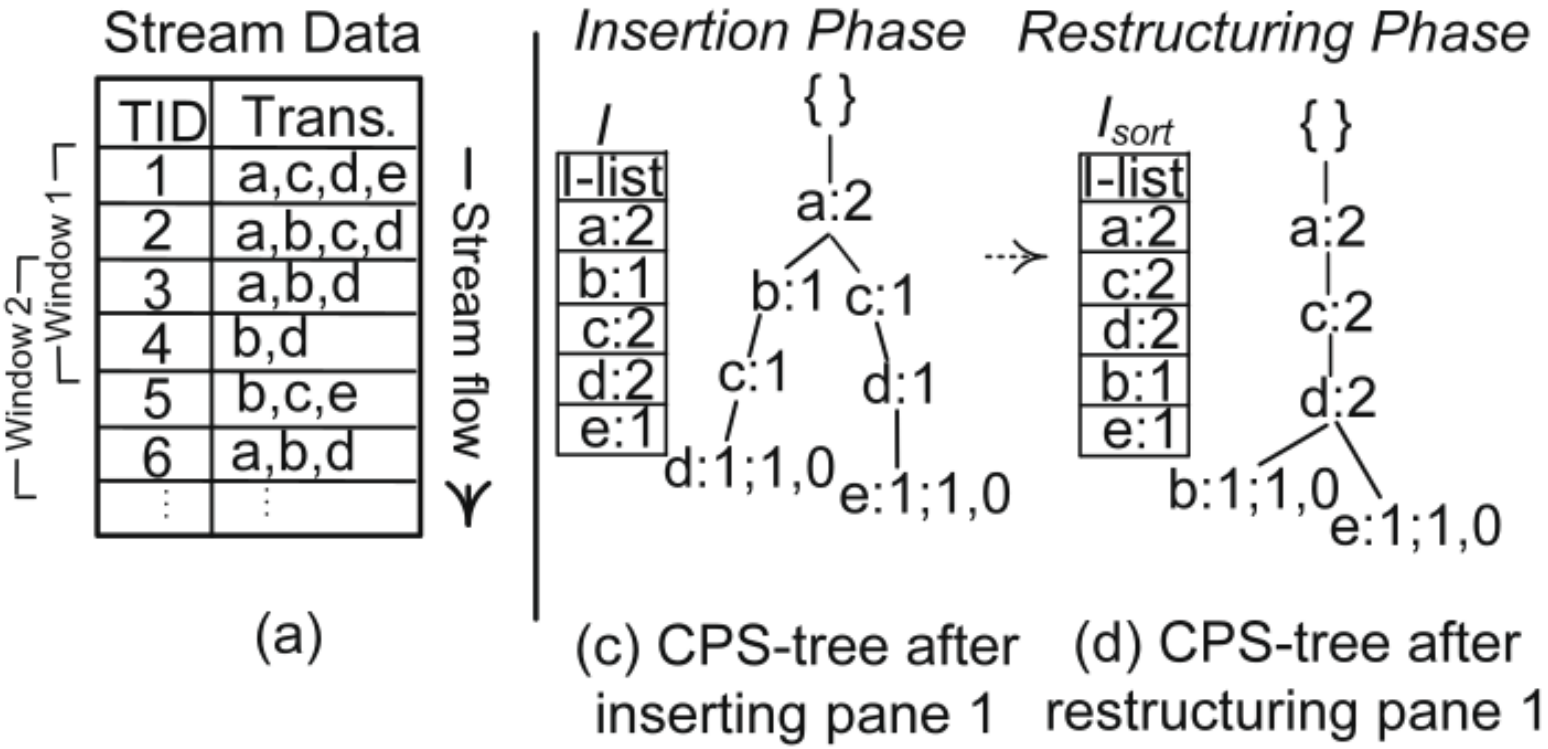
```
    Begin
1:      w ← ∅;
2:      T ← a prefix-tree with null initialization;
3:      Current_Sort_Order ← Initial_Sort_Order;
    //For the first Window
4:          While (w ≠ Window_size) do
5:              Call Insert_Pane(T);                                  // Insertion Phase
6:              Current_Sort_Order ← Frequency-descending sort order; // Restructuring Phase
7:              Restructure T;
8:              w = w+1;
9:          End While
    //At each slide of Window
10:         Repeat
11:             Delete the oldest pane information from T;            // Extracting the old pane
12:             Call Insert_Pane(T);                                  // Insertion Phase
13:             Current_Sort_Order ← Frequency-descending sort order; // Restructuring Phase
14:             Restructure T;
15:         End
    End

Insert_Pane(Tr)
    Begin
1:      p ← ∅;
2:      While (p ≠ Pane_size) do
3:          Scan transaction from the current location in Stream_data;
4:          Insert the scanned transaction into Tr according to Current_Sort_Order;
5:          p = p+1;
6:      End While
    End
```

Fig. 3. The CPS-tree construction algorithm.

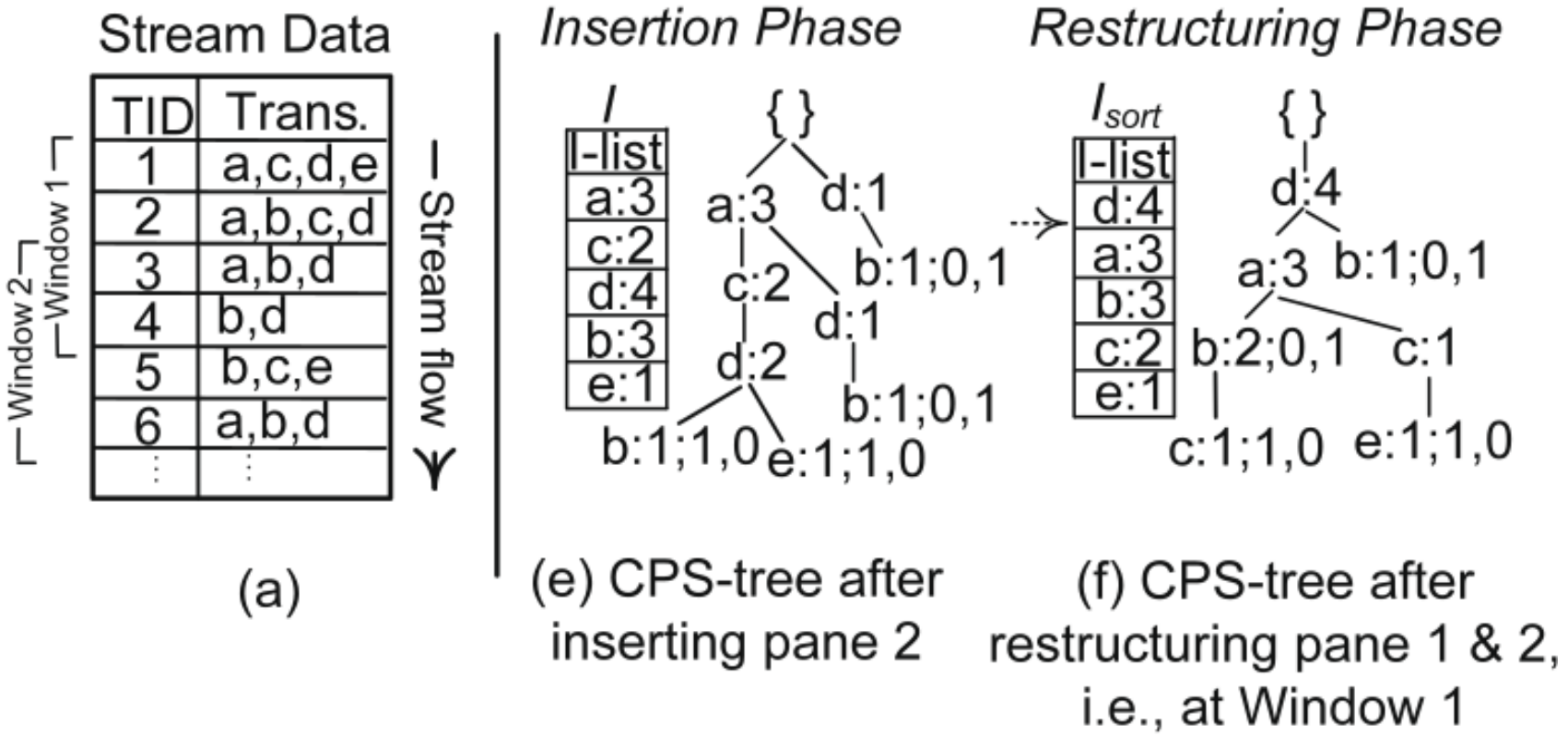
CPS-tree construction



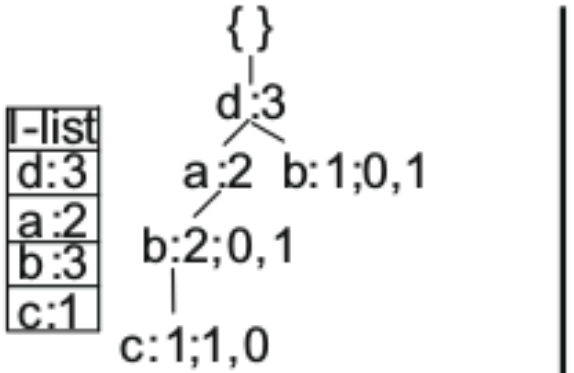
CPS-tree construction



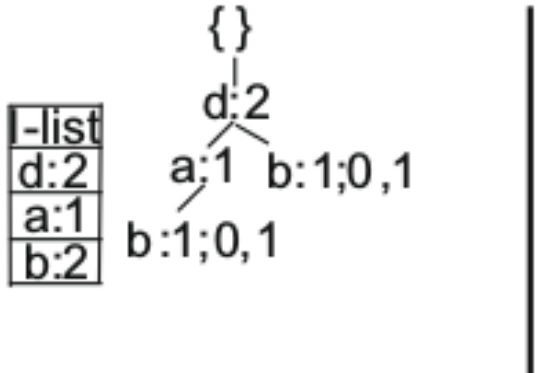
CPS-tree construction



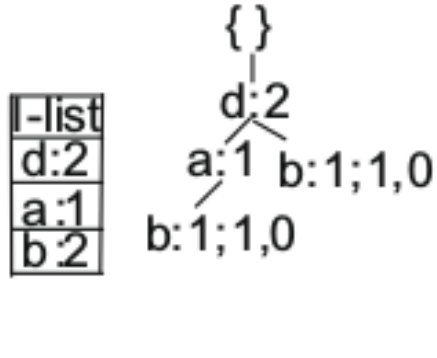
CPS-tree construction



(a) After processing for '*e*'

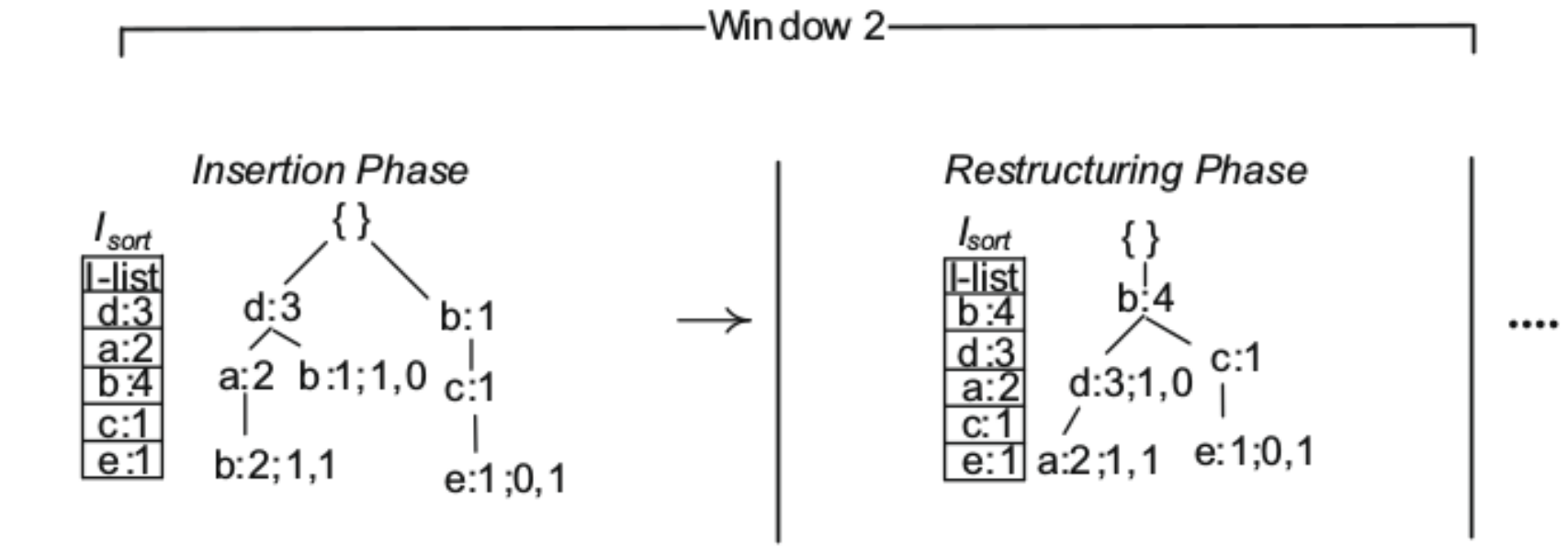


(b) After processing for '*c*'



(c) After processing for '*b*', '*a*', and '*d*'

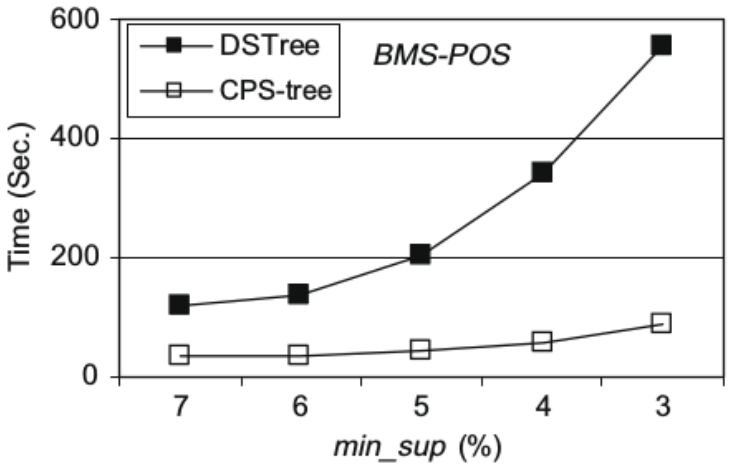
CPS-tree construction

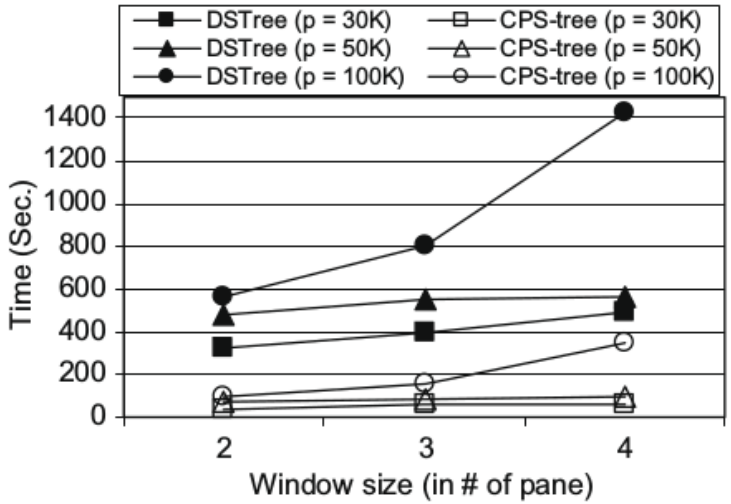


(a) CPS-tree after inserting new pane (i.e., *tids* 5 and 6) at Window 2

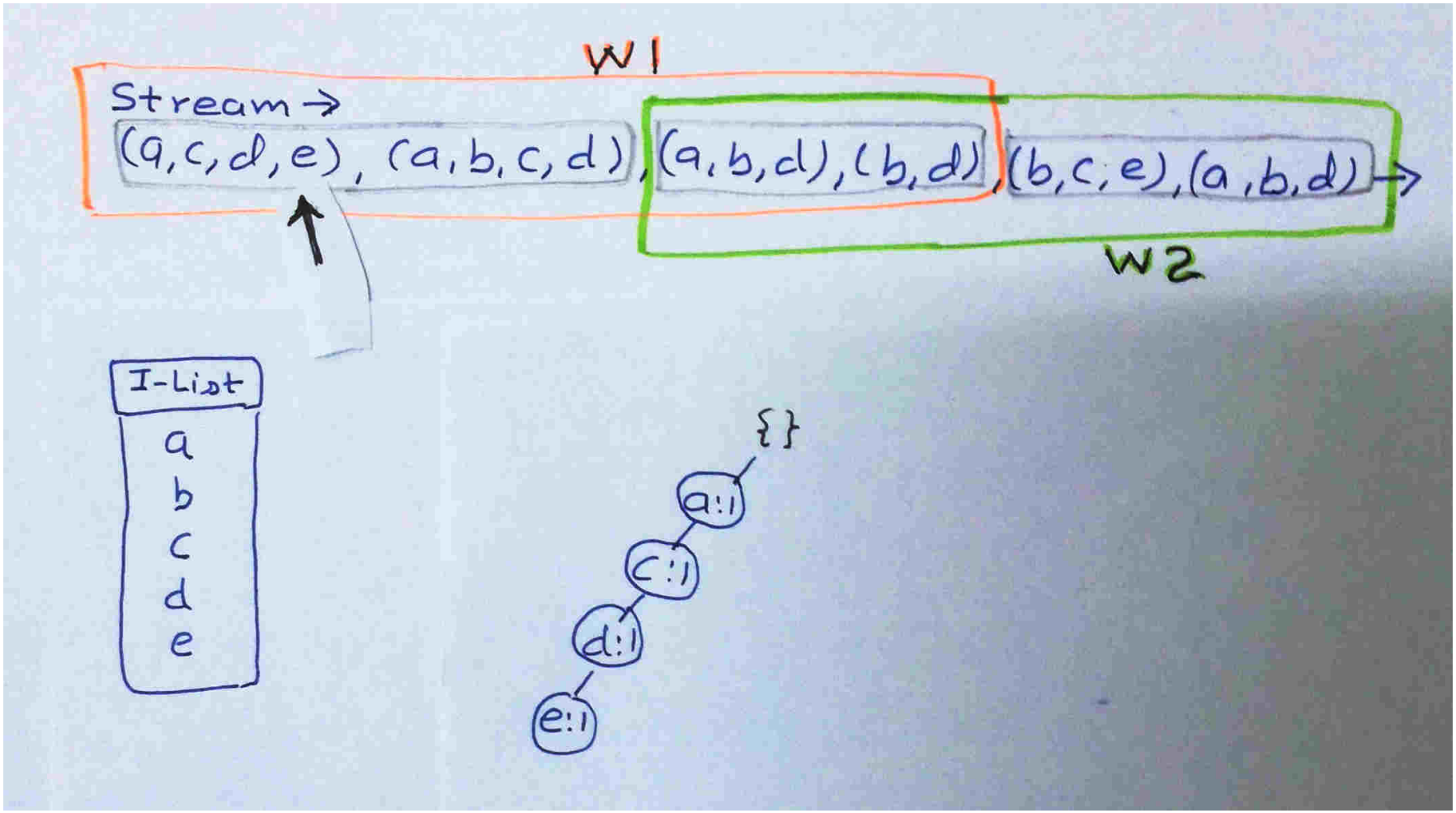
(b) CPS-tree after restructuring at Window 2

CPS-tree Performance

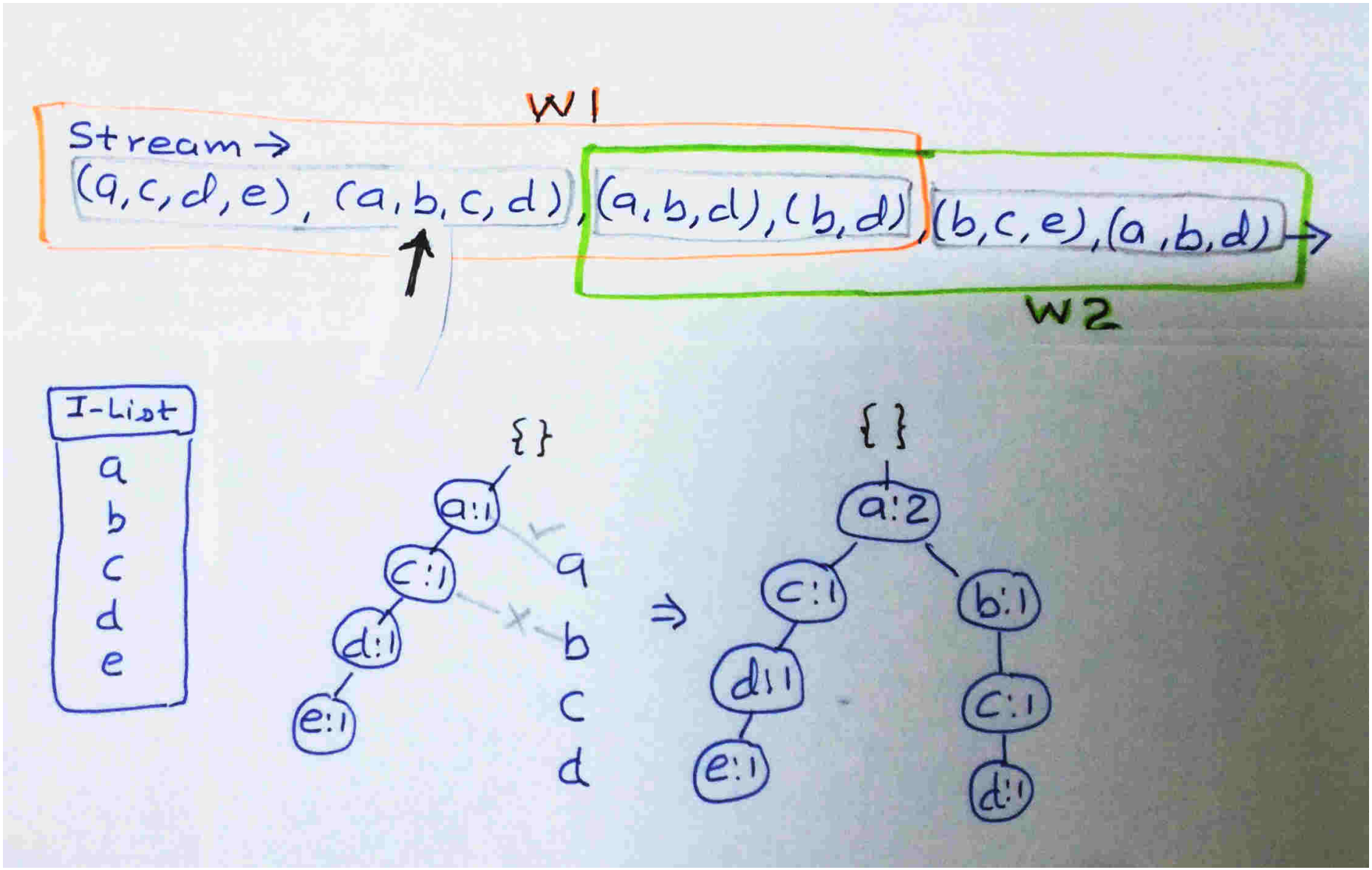




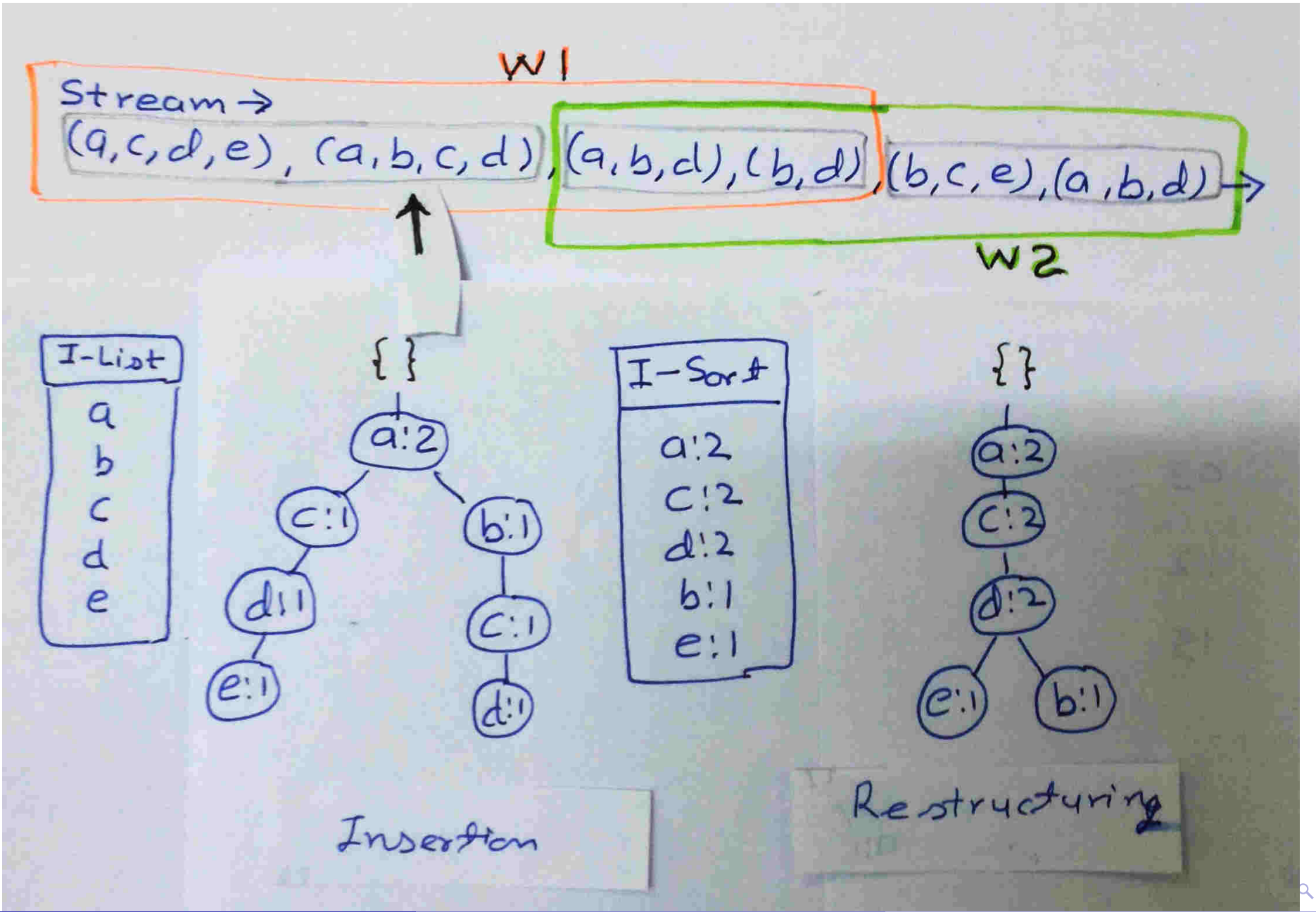
CPS-tree construction



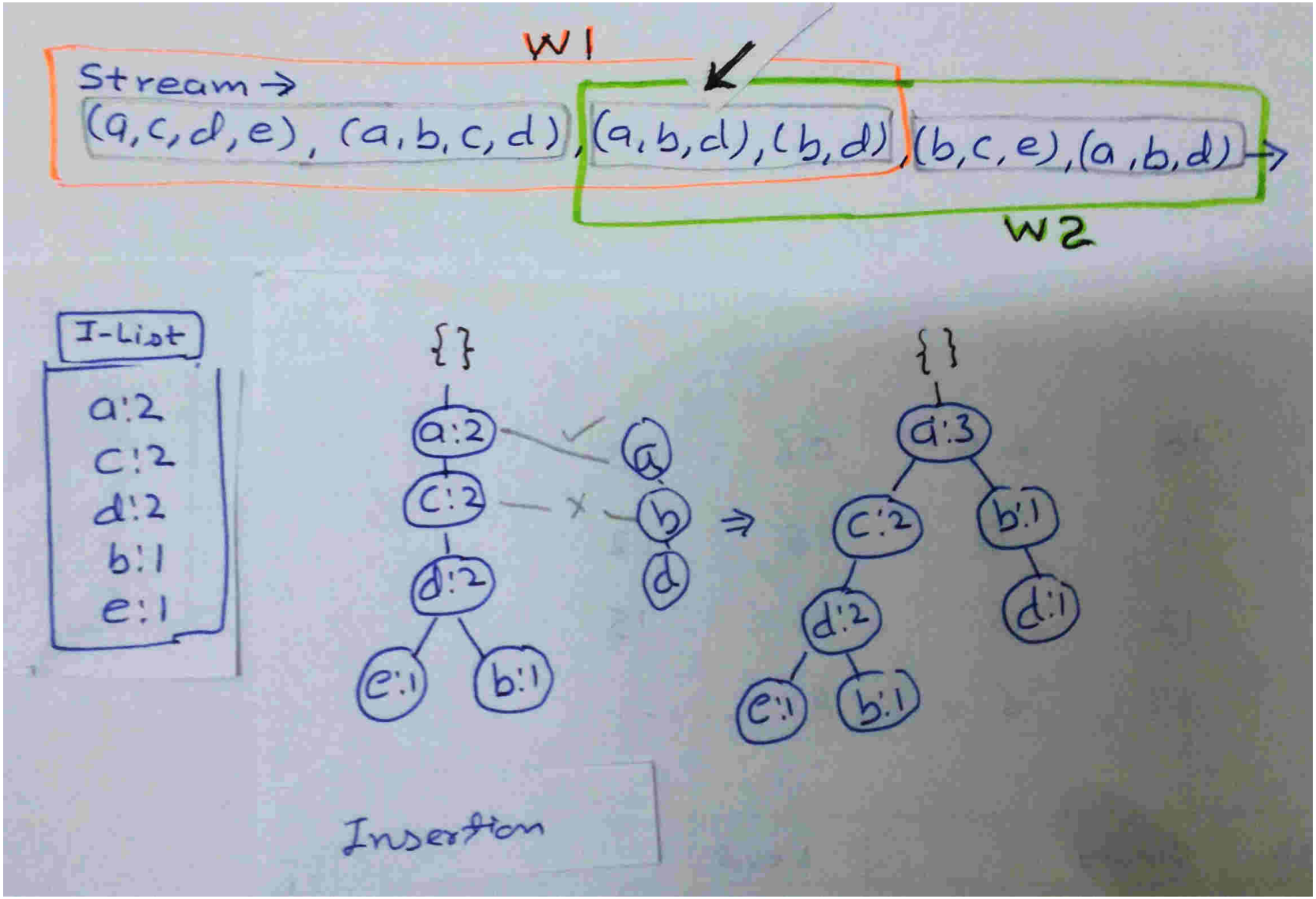
CPS-tree construction



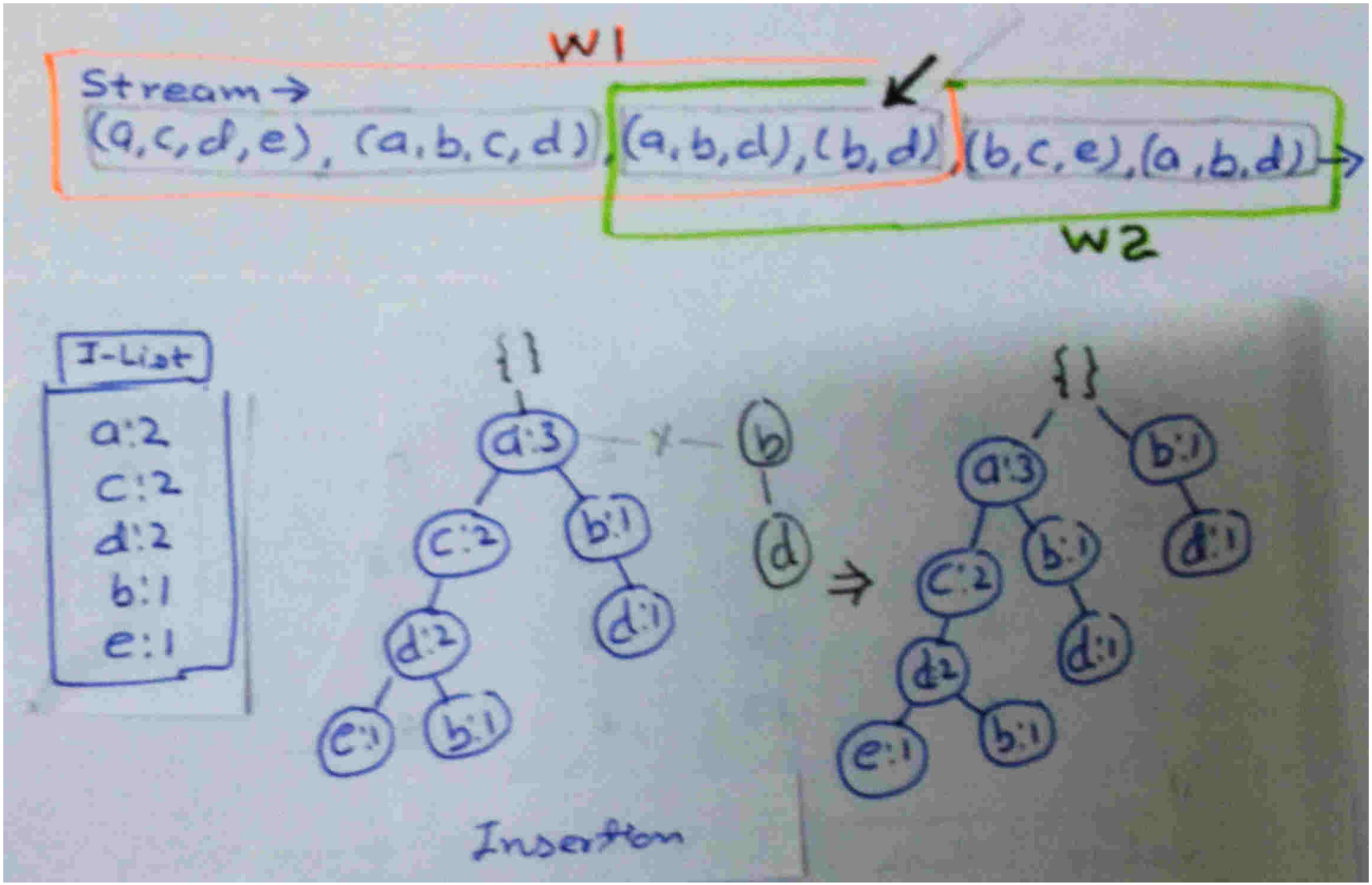
CPS-tree construction



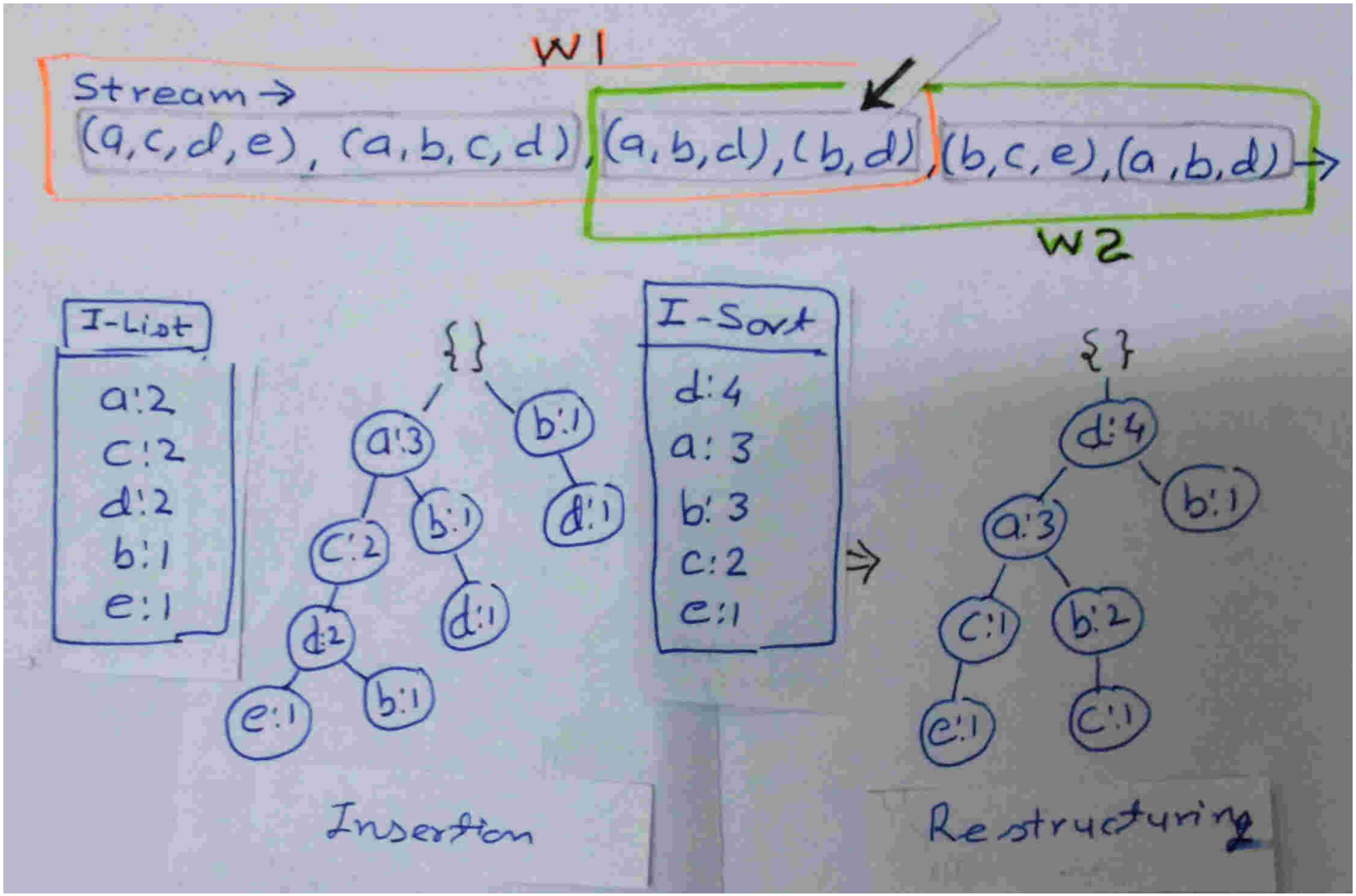
CPS-tree construction



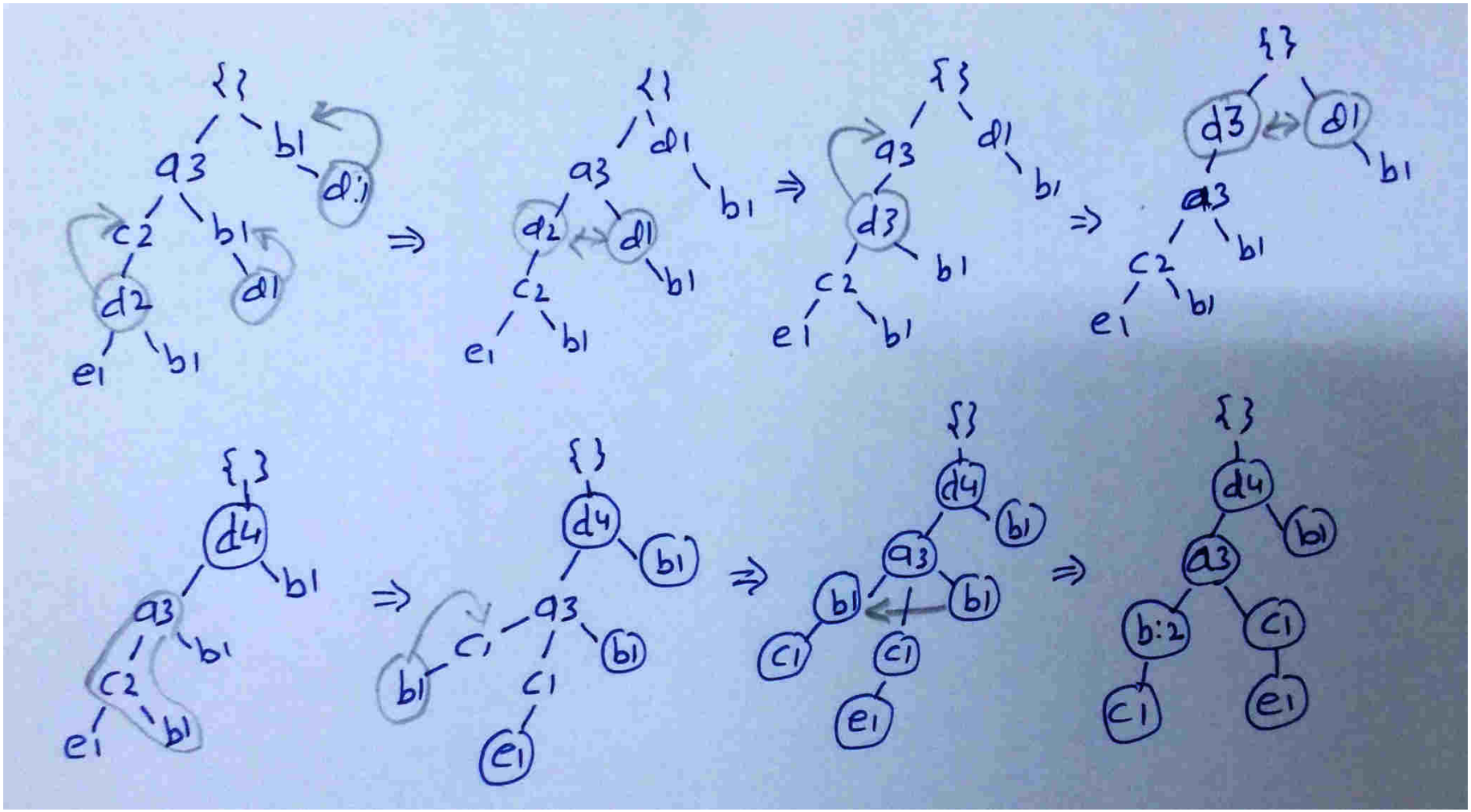
CPS-tree construction



CPS-tree construction

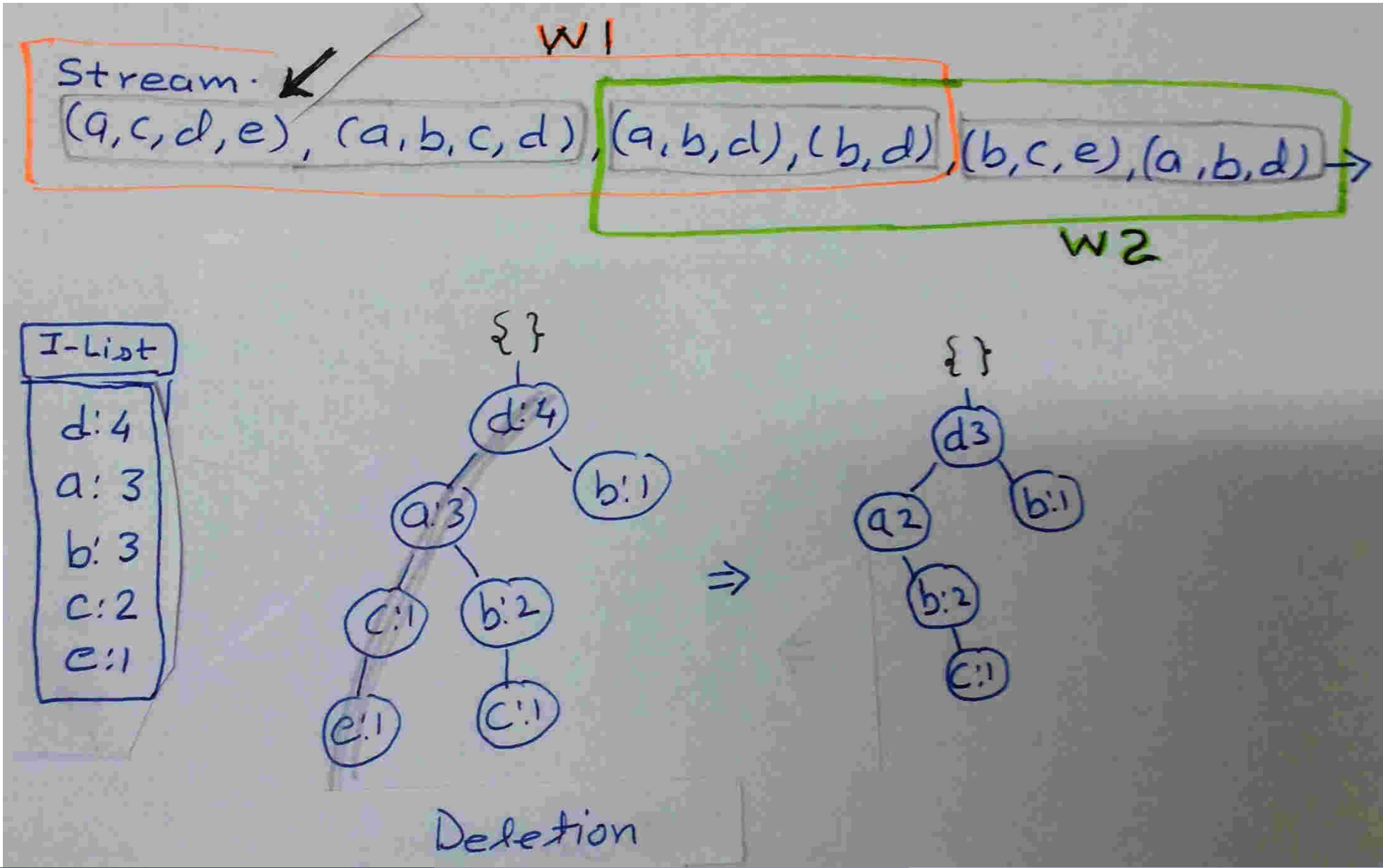


CPS-tree construction

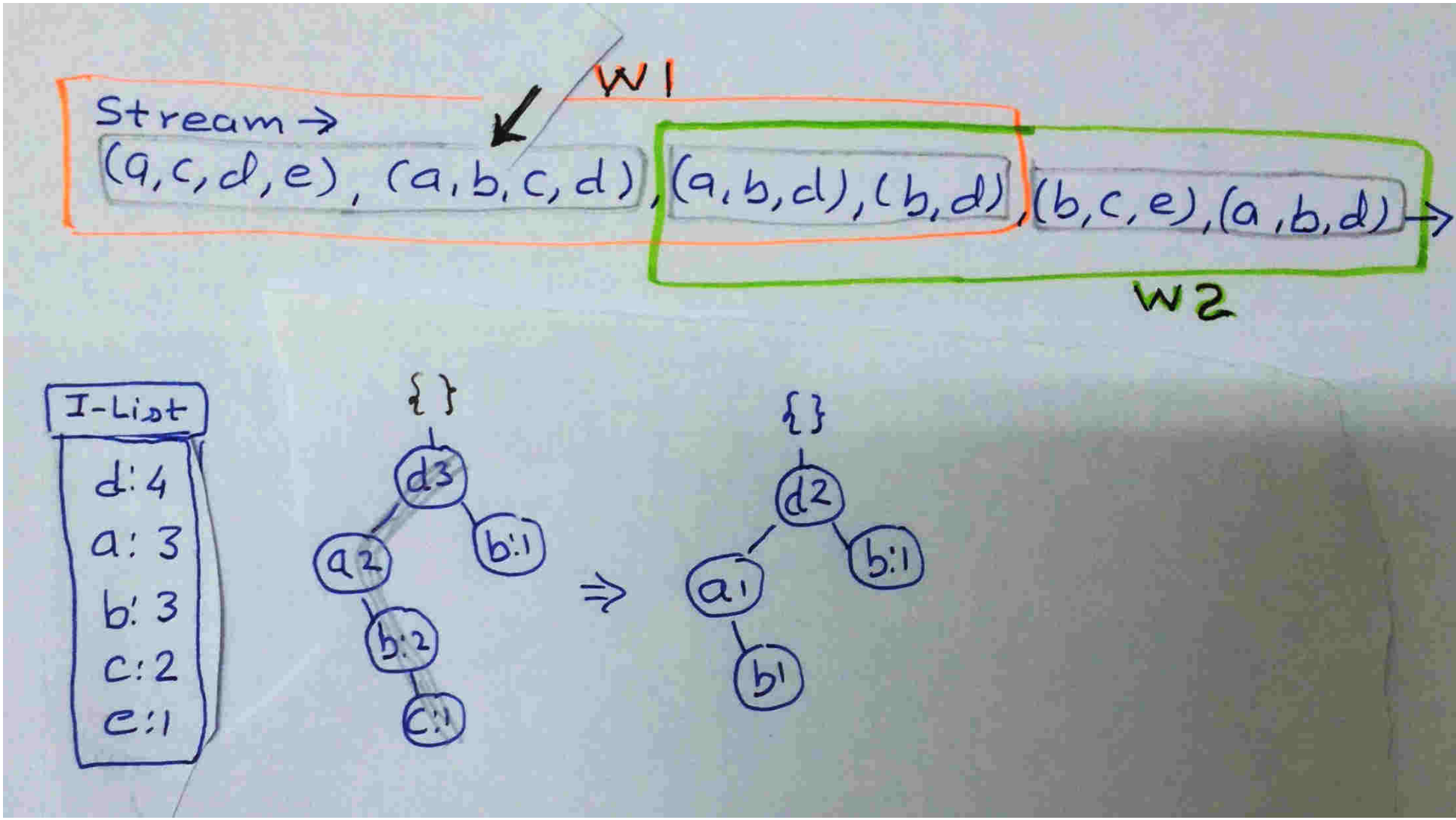


I-Sorted: d,a,b,c,e

CPS-tree construction

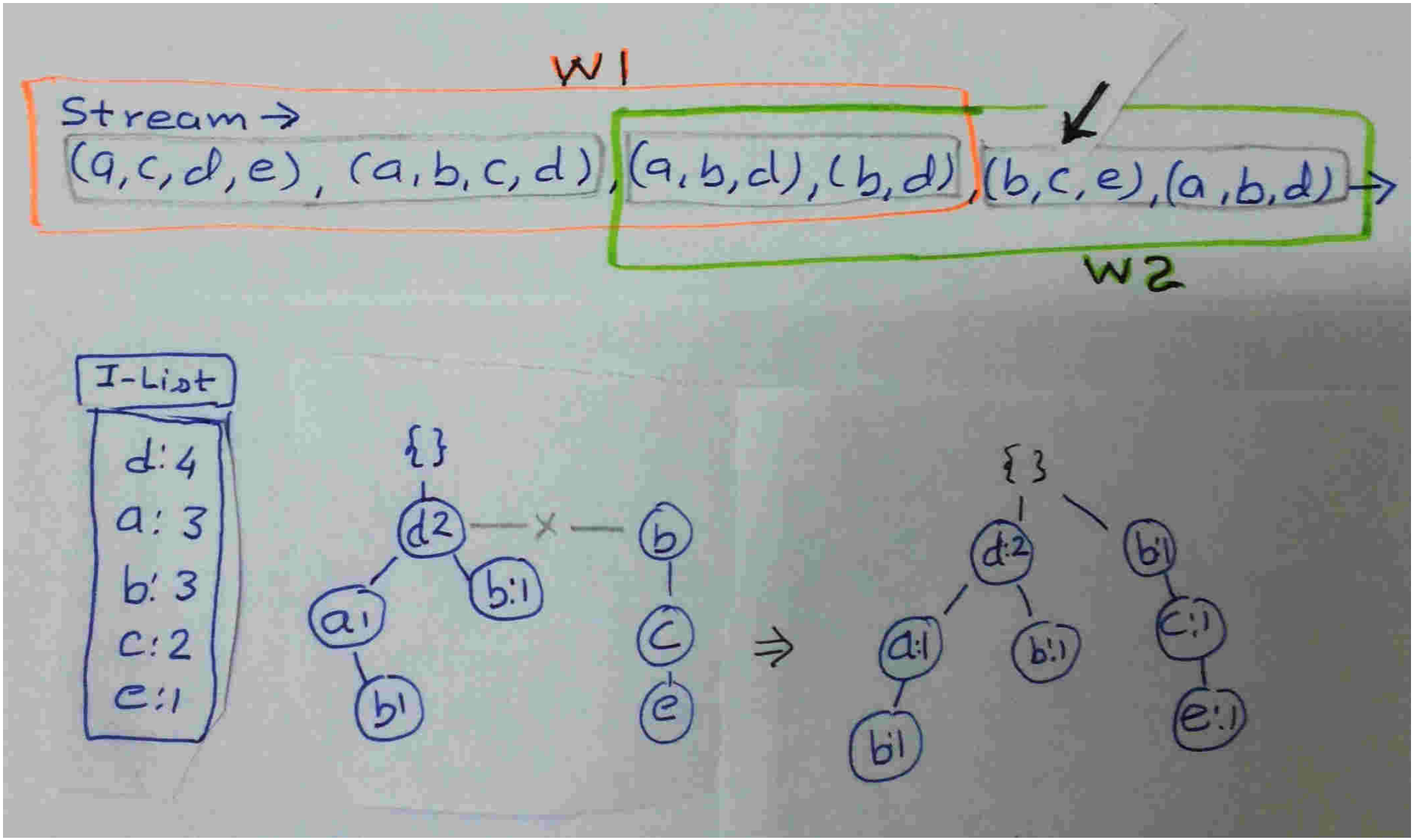


CPS-tree construction



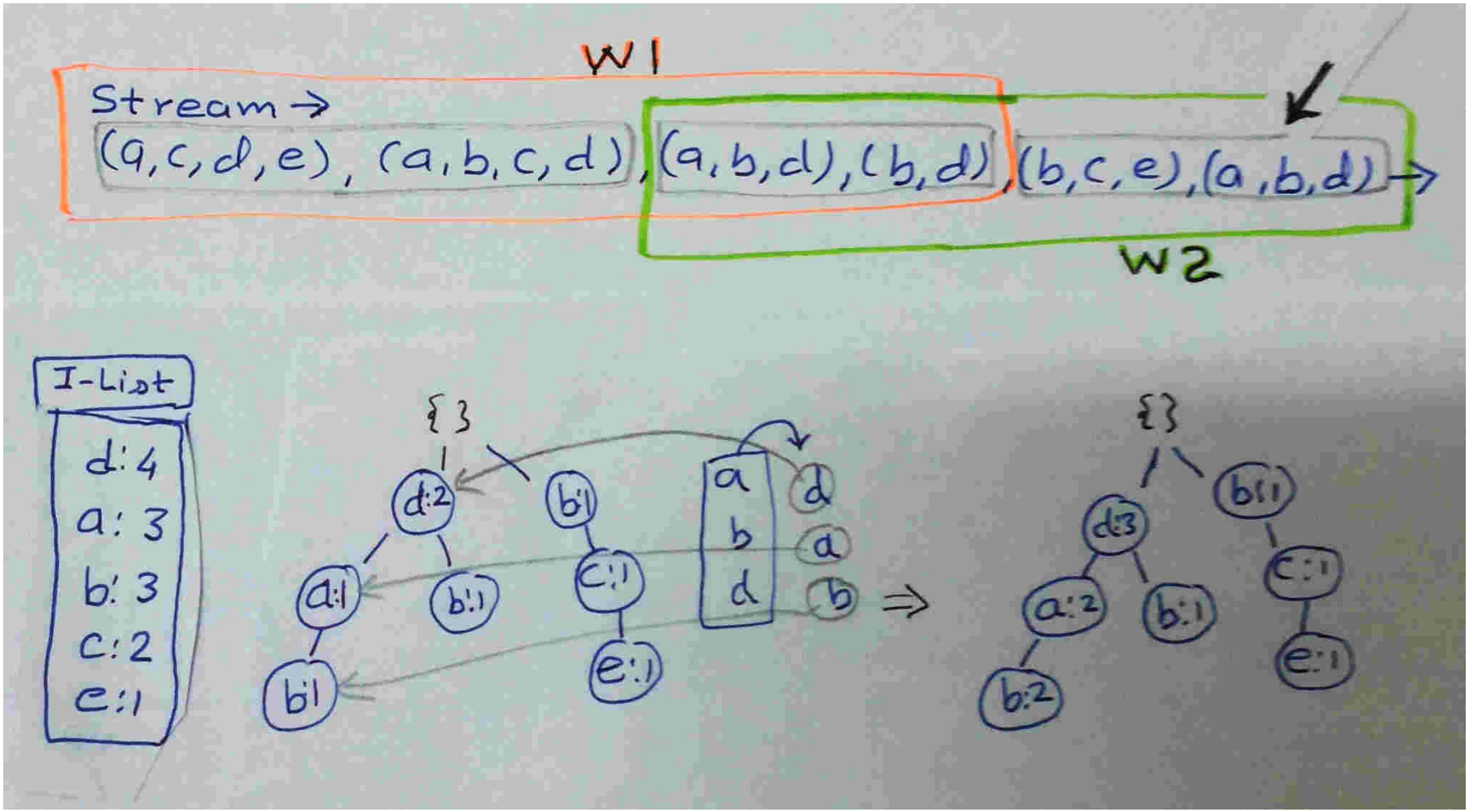
I-Sorted: d,a,b,c,e

CPS-tree construction



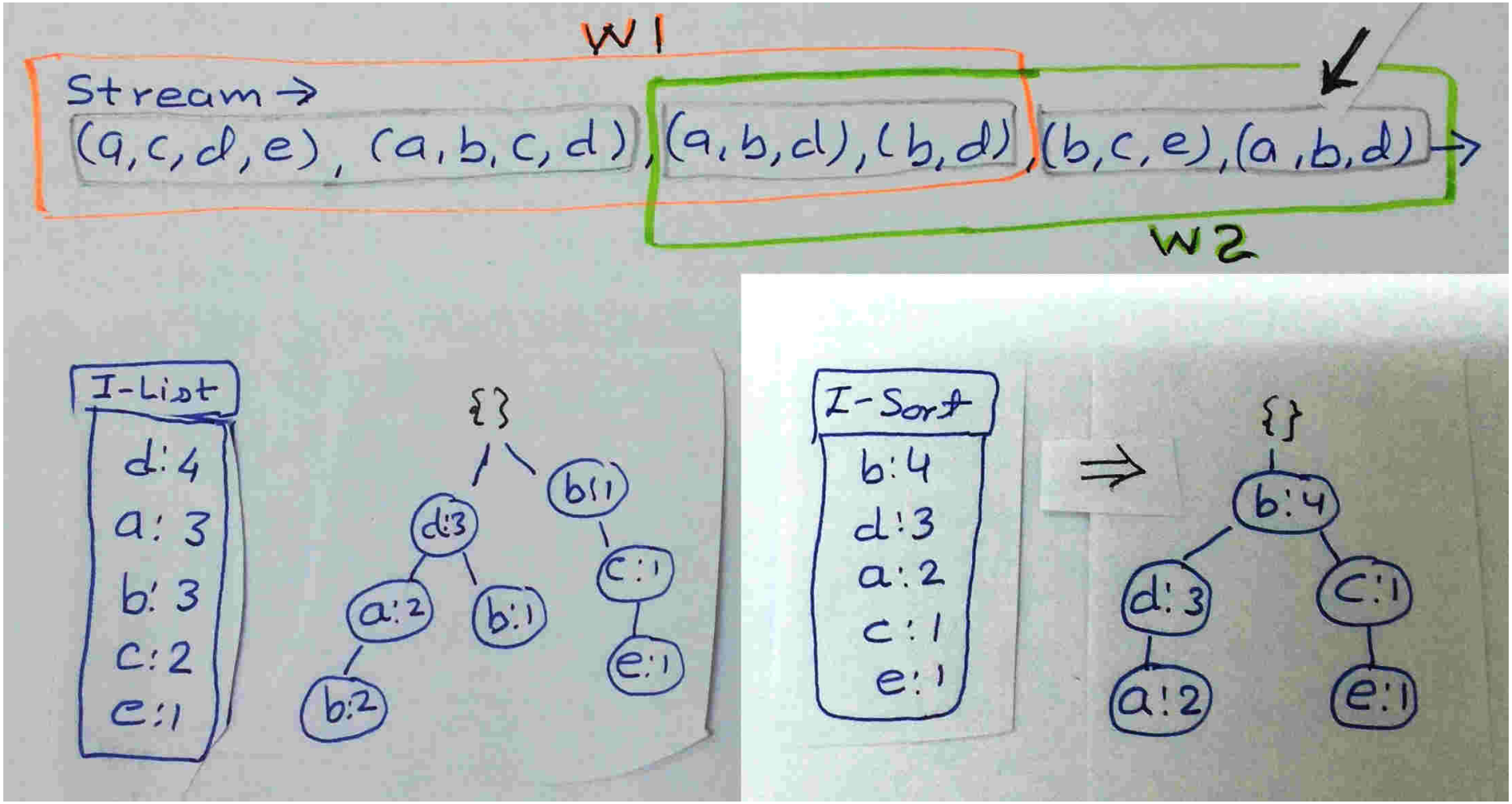
I-Sorted: d,a,b,c,e

CPS-tree construction



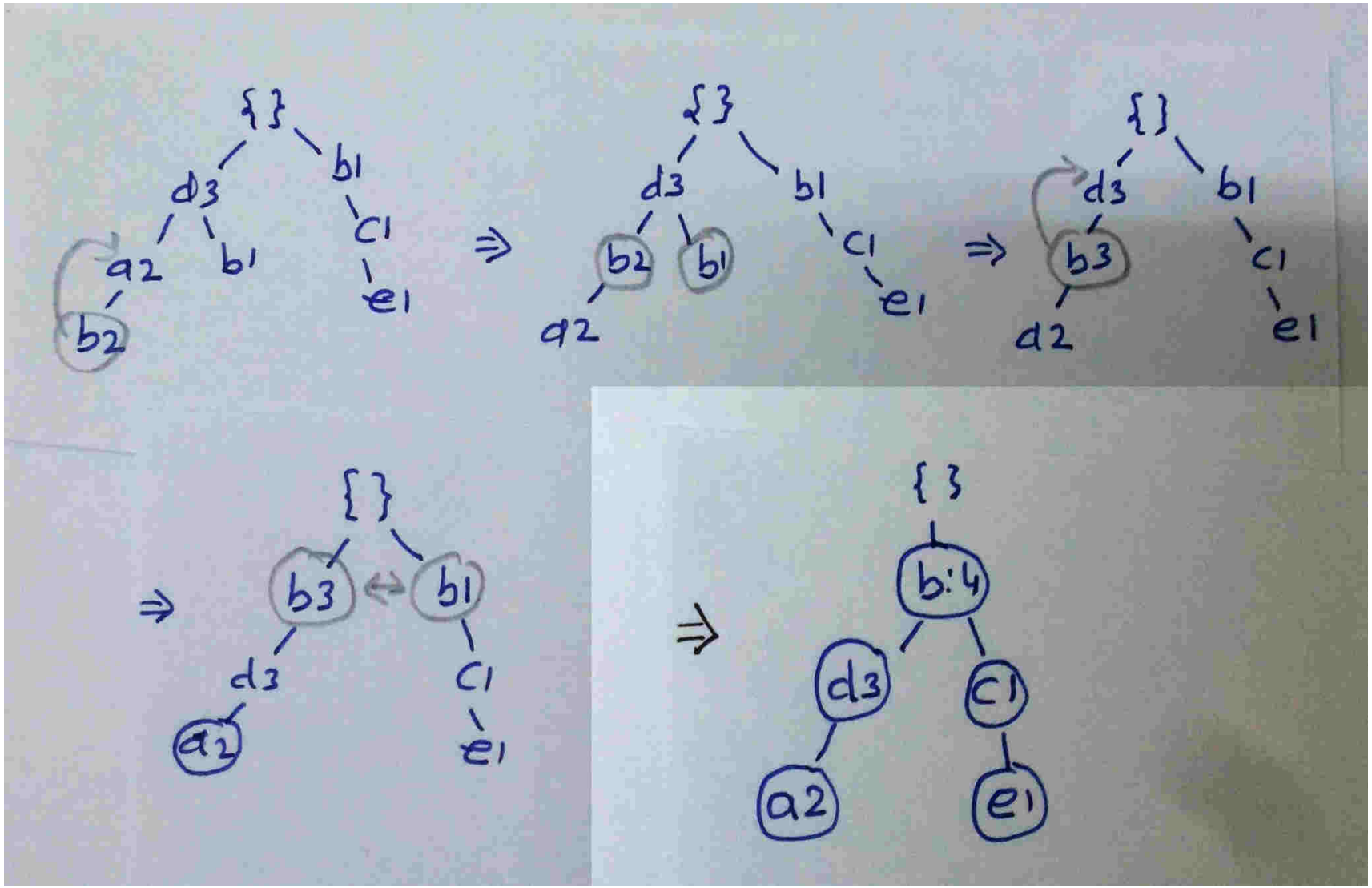
I-Sorted: d,a,b,c,e

CPS-tree construction



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

CPS-tree construction



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

Thank You!

Thank you very much for your attention!

Queries ?