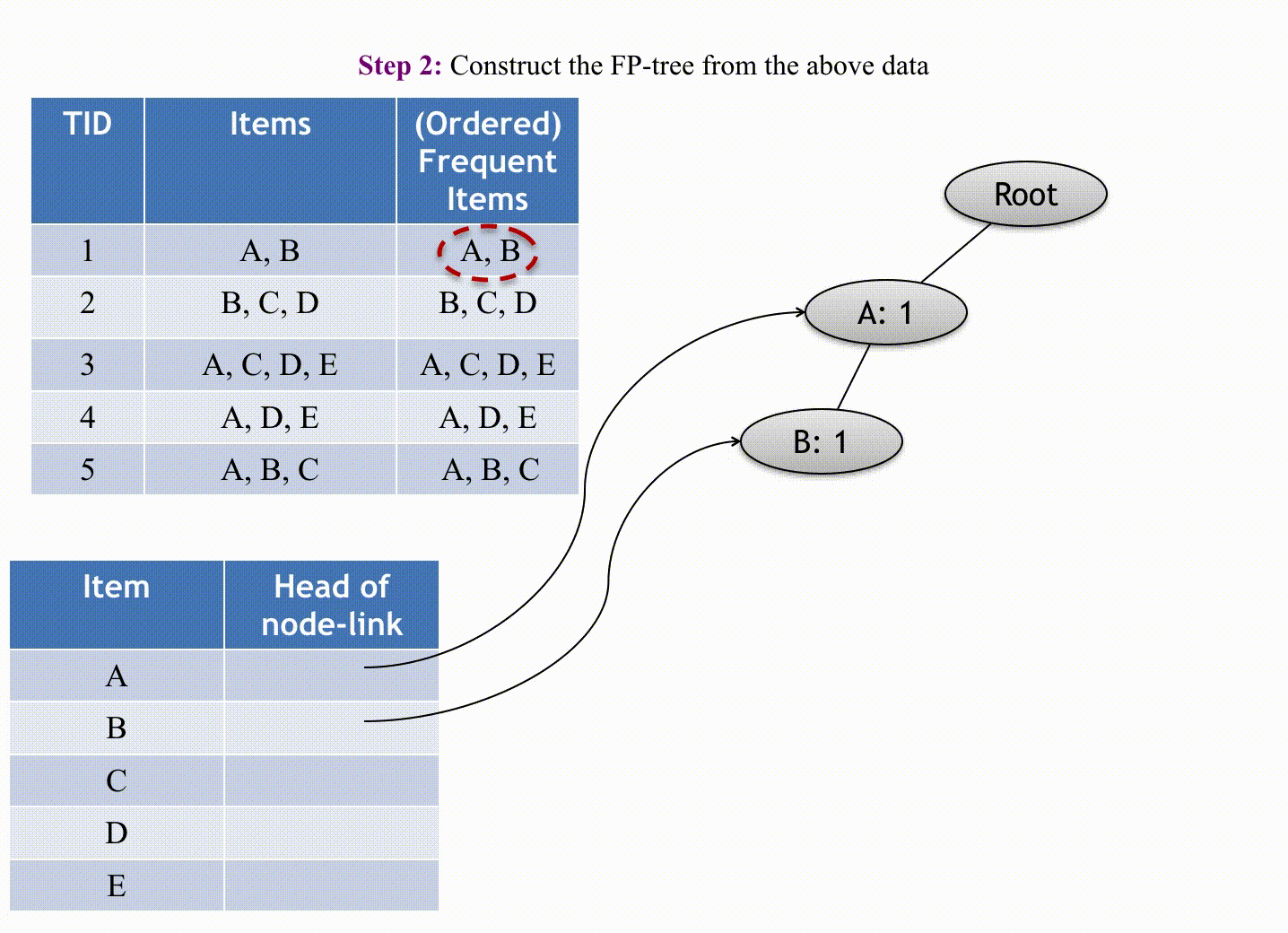
For this project, after reading data again with the previous approach instead of grouping items as preprocess we removed data from 10 to 19. Then I made a unique set of data’s within a transaction. I did this so I count my items more easily. Then I made a Node class in which it has name of the item (0-9) and count of the item and a link from header table to the tree:



Then I started to create tree. First, I counted the frequency of headers (0-9) in dataset and printed them.

Since they had 512 frequency data I decided to run the algorithm for first 10 item as well so we can see and measure the effect of algorithm:

For all items:

print(header) # {0: 512, 1: 512, 3: 512, 4: 512, 5: 512, 6: 512, 7: 512, 8: 512, 2: 512, 9: 512}

For 10 first items:

{0: 7, 1: 2, 3: 4, 4: 8, 5: 4, 6: 3, 7: 5, 8: 3, 2: 2, 9: 2}

And chosed to have a 4 threshold. Second, I removed the headers that have below threshold data. Then, I initialized my tree and root node and started the second round going through the data. In this round, at first I wrote the code for not being able to find a node in header in which I just added node to the root. But when we can traverse from root with respect to the highest frequency in data I traversed tree in the existing path to add it to the tree when we cannot traverse the tree anymore and in my way I updated each value by count and incremented our node value. At last for showing data I indented data in amount of the level of the child so we can traverse all child from roots and create tree.

root 1

4 8

0 5

7 3

3 2

5 1

5 1

7 2

3 1

5 1

3 1

5 1

0 2

Root has a child 4 with frequency of 8 then is 40 with frequency of 5 and after traversing a while 4075 has frequency of 1 like 4073.