Improved project summary

During the past busy week, I am concentrating on the code improvement by the feedback from the last technical interview. Thanks to the nice comment from Guanglin and Daniel, which give me the inspiration and guide to let my code much better than before.

Due to the time limitation, I divide the improvements into two parts and upgrade into two versions respectively. The first version focuses on *get_list* function, which I implement a nested dictionary instead of putting each dictionary of price list in a list. There is a significant change in the second version, which I implement the *Trie* data structure to finish the match between the longest prefix and input telephone number.

The logic of the first version is almost the same with the last week version. As for the nested dictionary, although there is only one line code:

dictionary.setdefault(key_a, { })[row[0]] = row[1]

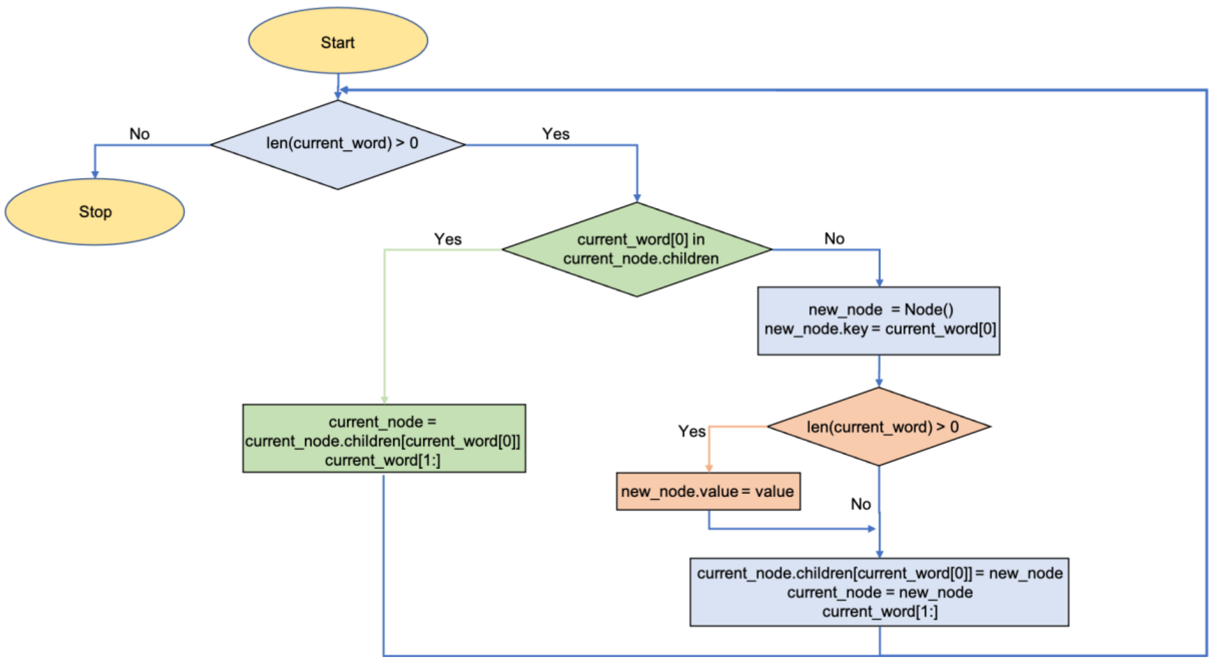
it took me a lot of time by searching the solution on the website and test. And the different in *find_price* function is that when we try to match the prefix, we need to get the value twice in to for loop:

dictionary.get(company).get(phone_number)

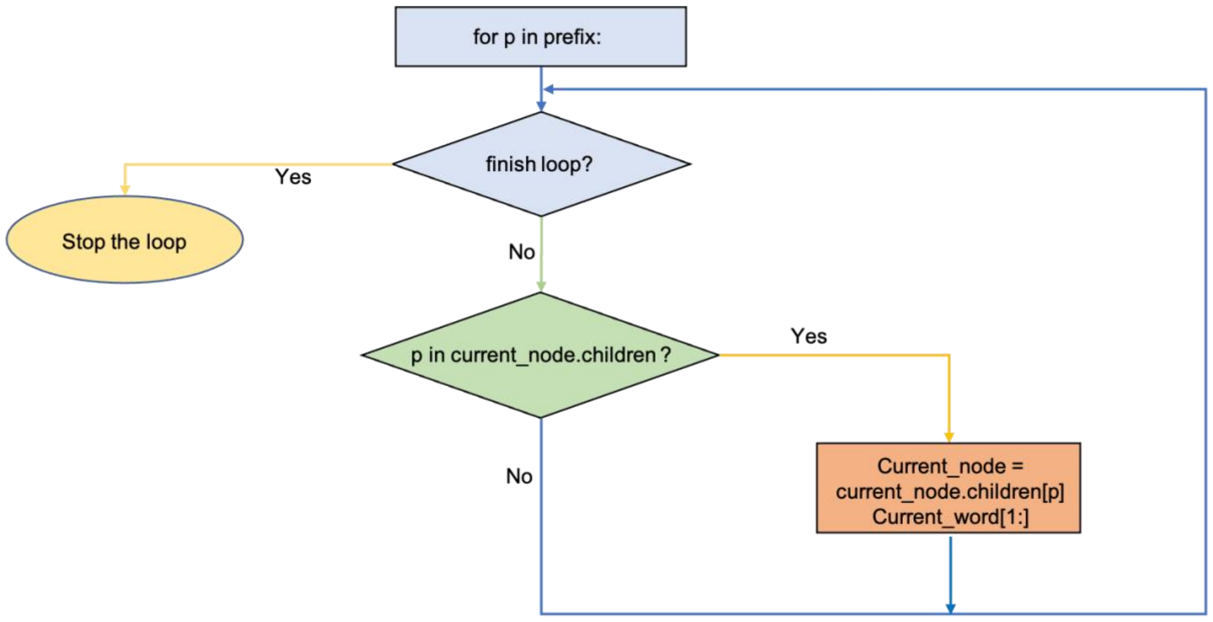
and when we finish the search, I use the del function to delete the current company dictionary instead of *.remove* function.

As for the second version of improvement, I dropped the previous find_price function, and implement a Trie data structure. Firstly, I created a Node class with the key, value and children. Secondly, I created the Trie class, which inheritance from the Node class, followed by the main functions in the second version:

- *insert* function: there are two values to input into the function, one is prefix, and another is value, which the dictionary of price list will be insert to the tree, and each digit of prefix are the children node of root. And the corresponding price will be the leaf of the last digit of each prefix. Followed by the flow chart demonstrate the logic of insert function:


Start
No
len(current_word) > 0
Yes
Stop
Yes
current_word[0] in current_node.children
No
new_node = Node()
new_node.key = current_word[0]
current_node = current_node.children[current_word[0]]
current_word[1:]
Yes
len(current_word) > 0
new_node.value = value
No
current_node.children[current_word[0]] = new_node
current_node = new_node
current_word[1:]

- *search* function: we need to input one value into the function – telephone number. Firstly, we initial the *current_work* as the input, and *current_node* as the root. Then we can check if the telephone number start with the existing prefix by the for loop, it can traversal each digit of telephone number and find the longest match. Finally, the last node will be kept, and the corresponding price is the value (leaf) of the last node. If there is no value of the last node, it means that there is no matched prefix and *return False*, or print the corresponding price and *return True*. Followed by the flow chart illustrate the logic of the search function:


for p in prefix:
finish loop?
Yes
Stop the loop
No
p in current_node.children ?
Yes
No
Current_node = current_node.children[p]
Current_word[1:]

Finally, although I improved my solution, there is also a space for improvement if I have more time to debug the code.

- Firstly, it should be the combination of above two improvement points, since the second version of improvement is also used the dictionary list instead nested dictionary to store the price list from each company. It will be more professional by using the loop to build the tree from the nested dictionary.
- Secondly, the current code can just find the corresponding price with the longest prefix in each company, it will take more time to implement a function to find the cheapest price among different operators.
- Thirdly, I have no idea about how to set a name for each tree, that is why I can just find the price in each operator without the operator's name.

Although the past week is also a tough week for me, the process is a precious experience and practice for me the upgrade my coding skill, which I can improve myself even a little bit every day. It is highly appreciated that I can have the further comments and feedback!

Reference list

1. The nested dictionary: https://stackoverflow.com/questions/25833613/safe-method-to-get-value-of-nested-dictionary
2. Trie data structure: https://brilliant.org/wiki/tries/