Python for Data Science SP1 2023/2024
Assignment 5

There is only 1 problem in this assignment.
To submit:

- A python (.py) file with clearly marked sections and comments with the solution to the problem. The example cases in this document should be included at the bottom of the file.

Do not forget to add your name and your group partner's name at the very top of your file.
Note:

- Think about different inputs which might help you assure that your implementation is correct.
- If you add in any details or assumptions, please describe these in comments in your submission.

1 Tries

A Trie is a type of tree data structure that is used for locating specific keys from within a set. Tries are commonly used in predictive text or autocomplete of words, because of their ability to store prefixes of words. They are also used in web search engines where they store collections of searchable words, with values being URLs.

Consider a Trie as given in Figure 1. We have a root node which is blank. Then by following the letters in a word, we end up at a destination node, which contains the last letter of the word. If the word is complete the final letter node will be given a value.

The data structure will have the following fields and init function:

```
from typing import Optional, Dict
class Trie():
    def __init__(self) -> None:
        self.value: Optional[Value] = None
        self.endOfWord: bool = False
        self.active: bool = True
        self.children: Dict[Key, Trie] = {}
```

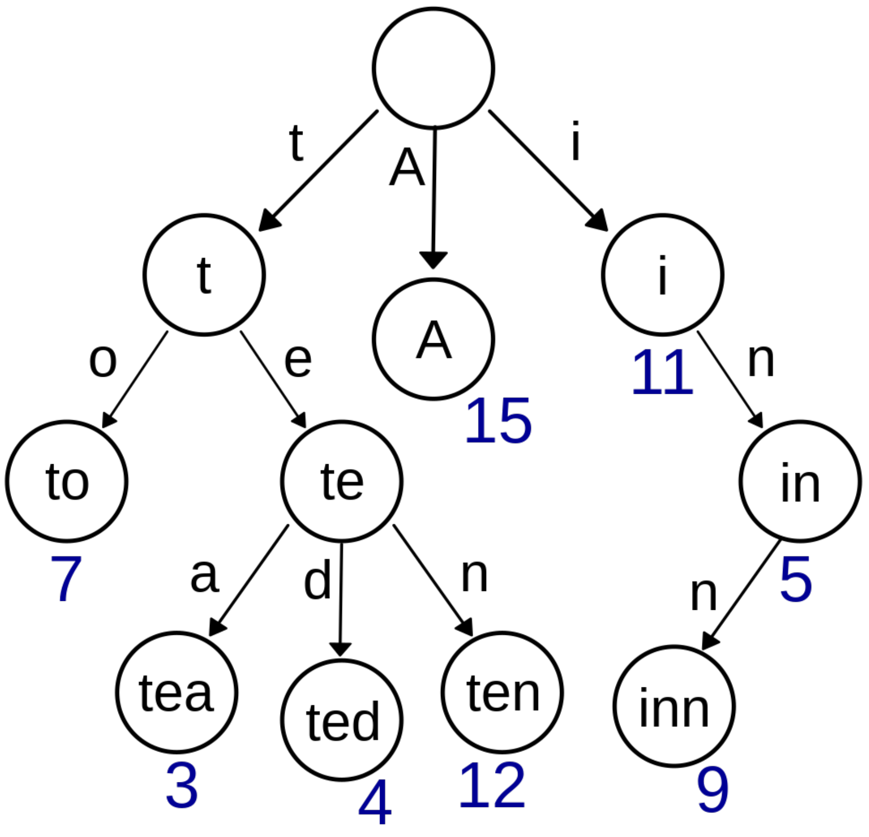

Figure 1: Trie Example, from wikipedia

The `self.value` store the value associated with a particular node, the type of `Value` in the first part of this problem will be `int`. The `self.endOfWord` field marks if the current node is the end of a word. The field `active` is used for implementing delete. If a note is marked as inactive it can be deleted. Finally, the `self.children` field stores the child nodes of the current node, these child nodes are stored in a dictionary with keys or type `Key`, which in the first part of this assignment will be of type `str`.

Your tasks are as follow.

1. Implement a method called `insert` on the Trie class that inserts a new word into the trie. Implement a method called `fetch` on the Trie class that returns `None` if the word is not in the Trie, and the `value` of the word if it is in the Trie. The function signatures should be:

```
def insert(self, path:str, val:int) -> None
def fetch(self, path:str) -> Optional[int]
```

The following code should run:

```
t = Trie()
for word,val in zip(["alphabet", "beta", "gamma"], [1,2,3]):
    t.insert(word, val)

t.fetch("alpha") # None
t.fetch("beta") # 2
t.fetch("alphabet") # 1
t.fetch("Gamma") # None
t.fetch("gamma") # 3
```

2. Implement a method called `delete` on the Trie that removes a given word from the Trie if and only if it exists in the Trie and returns its value, otherwise it returns None. The function signature should be:

```
def delete(self, path:str) -> Optional[int]
```

The following code should run:

```
t = Trie()

t.insert("alpha", 10)
a = t.delete("alpha") # a == 10
t.fetch("alpha") # None

t.insert("alpha", 10)
t.insert("alphabet", 22 )
a = t.delete("alpha") #a = 10
t.fetch("alphabet") # 22
t.fetch("alpha") #None
```

3. The key and value types above were of type string and int respectively. But the Trie is a generic data structure, where the Key and Value can be of many different types. Update your Trie data structure to have file paths as the Key and strings of dates of the last time the file was modified as the value. The file path will be given as a collection of strings. As an example path:

```
path = ["/", "home", "/", "user", "/", "folder1", "/", "filename"]
```

And the dates will be given as `"2023-08-17"`. The following code should run:

```
t = Trie()

path1 = ["/", "home", "/", "user", "/", "folder1", "/", "file1"]
```

```
path2 = ["/", "home", "/", "user", "/", "folder1", "/", "file2"]
path3 = ["/", "home", "/", "user", "/", "folder2", "/", "file3"]

t.insert(path1, "2022-09-22")
t.insert(path2, "2023-01-19")
t.insert(path3, "2021-04-01")

t.fetch(path1) # 2022-09-22
t.delete(path1) # 2022-09-22
t.fetch(path2) # 2023-01-19
t.fetch(path1) # None
t.fetch(["/", "home"]) # None
t.fetch(path3) # 2021-04-01
```