İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

FILE ORGANIZATION

LAB APPLICATION 10.05.2024

MUHAMMET TALHA ODABAŞI

1306220012

1. BINARY TREE

```python
class Node:
    def __init__(self, data, file):
      self.left = None
      self.right = None
      self.data = data
      self.file = file

    def insert(self, data, file):
        if self.data:
            if data < self.data:
                if self.left is None:
                    self.left = Node(data, file)
                else:
                    self.left.insert(data, file)
            elif data > self.data:
                if self.right is None:
                    self.right = Node(data, file)
                else:
                    self.right.insert(data, file)
            else:
                self.data = data

    def PreorderTraversal(self, root):
        res = []
        if root:
            res.append(f"{root.data} - {root.file}")
            res = res + self.PreorderTraversal(root.left)
            res = res + self.PreorderTraversal(root.right)
        return res

    def inorderTraversal(self, root):
        res = []
        if root:
            res = self.inorderTraversal(root.left)
            res.append(f"{root.data} - {root.file}")
            res = res + self.inorderTraversal(root.right)
        return res

    def PostorderTraversal(self, root):
        res = []
        if root:
            res = self.PostorderTraversal(root.left)
            res = res + self.PostorderTraversal(root.right)
            res.append(f"{root.data} - {root.file}")
        return res

    def search(self, root, data):
        if root is None or root.data == data:
            return root
        if root.data < data:
            return self.search(root.right, data)
        return self.search(root.left, data)
```

As you can see on the right side I have created a binary tree with python which holds hash of absolute path and file name.

2. EXTRA FUNCTIONS

```
def create_tree():
    root = Node(500000, None)

    media_list = os.listdir()
    cwd = os.getcwd()
    for media in media_list:
        mini_media_list = os.listdir(os.path.join(cwd, media))
        for mini_media in mini_media_list:
            for file in os.listdir(os.path.join(cwd, media, mini_media)):
                as_int = hash_generator(os.path.join(cwd, media, mini_media, file))
                root.insert(as_int, file)
    return root
```

As you can see in the above function, we loop through files in Media_Root and its sub directories and creating a binary tree with MD5 hash value of file's absolute paths.

```
def hash_generator(file):
    a = md5(file.encode())
    b = a.hexdigest()
    as_int = int(b, 16) % 999983
    return as_int
```

And hash generator function is simple as you can see. Getting the value of file's absolute path as parameter and calculating its MD5 hash. Then to be able to store it in Integer value and to be able to perform comparison in binary tree, I am calculating modulo of biggest 6 digit prime number.

3. MAIN CODE

```
def menu():
    os.chdir(os.path.join(os.getcwd(), "Media_Root"))
    print("1. Insert")
    print("2. Delete")
    print("3. Tree Traversal")
    print("4. Search")
    choice = int(input("Enter choice: "))

    match choice:
        case 1:
            choice = int(input("[1] Music\n[2] Photos\n[3] Videos\nEnter Choise: "))
            match choice:
                case 1:
                    choice = int(input("[1] Pop\n[2] Jazz\nEnter Choise: "))
                    match choice:
                        case 1:
                            os.chdir(os.path.join(os.getcwd(), "Music", "Pop"))
                        case 2:
                            os.chdir(os.path.join(os.getcwd(), "Music", "Jazz"))
                    with open(f'track{len(os.listdir()) + 1}.mp3', 'w') as fp: pass
                case 2:
                    choice = int(input("[1] Events\n[2] Vacation\nEnter Choise: "))
                    match choice:
                        case 1:
                            os.chdir(os.path.join(os.getcwd(), "Photos", "Events"))
                        case 2:
                            os.chdir(os.path.join(os.getcwd(), "Photos", "Vacation"))
                    with open(f'photo{len(os.listdir()) + 1}.jpg', 'w') as fp: pass
                case 3:
                    choice = int(input("[1] Documentaries\n[2] Movies\nEnter Choise: "))
                    match choice:
                        case 1:
                            os.chdir(os.path.join(os.getcwd(), "Videos", "Documentaries"))
                            with open(f'doc{len(os.listdir()) + 1}.mp4', 'w') as fp: pass
                        case 2:
                            os.chdir(os.path.join(os.getcwd(), "Videos", "Movies"))
                            with open(f'movie{len(os.listdir()) + 1}.mp4', 'w') as fp: pass
        case 2:
            path = input("Enter the absolute path to the file: ")
            if os.path.exists(path):
                os.remove(path)
                print("File deleted successfully")
            else:
                print("File not found")
        case 3:
            root = create_tree()
            choice = int(input("[1] Preorder\n[2] Inorder\n[3] Postorder\nEnter Choise: "))
            ordered = []
            match choice:
                case 1:
                    ordered = root.PreorderTraversal(root)
                case 2:
                    ordered = root.inorderTraversal(root)
                case 3:
                    ordered = root.PostorderTraversal(root)
            for i in ordered:
                if i != "500000 - None":
                    print(i)
        case 4:
            root = create_tree()
            data = input("Enter the file path to search: ")
            hash_data = hash_generator(data)
            found = root.search(root, hash_data)
            if not found:
                print("File not found")
            else:
                print(f"File found: {found.file} - {found.data}")
        case _:
            exit()
```

All of the main function is on the right as you can see.

A menu is displayed to choose operations between Insertion, Deletion, Tree Traversal and Value searching.

According to user input we perform necessary functions.