CmpE 230 Project 2 - Duplicate Determiner
Student: Hazar ÇAKIR

Briefing:

I did this project in Linux system. All required arguments are implemented according to the description. Identic.py can be run from the console directly because I change the mod of file as executable. The project implemented in python 3.6.9. Program runs without any failure for all the cases that I faced. There is no special case that the program doesn't work as observed. Additionally it can work with both -d and -f arguments and scans both files and directories as wanted.

Code Analysis:

Source code is well commented. Every function and variable have names according to their function. Even so, I will examine my code elaborately. I believe there is no error in the code segments. Let me talk about libraries that I use first.

I use argparse library to handle arguments issue. With the help of argparse, all arguments situations handled easily and well.

I use os library to get directory lists, file sizes, to join directories, to determine some file is whether is a file or a directory and to get absolute path of some path.

I use hashlib library to hash values. I specifically use sha256() version as recommended in description file.

Now I want to talk about main dialect of my program. First of all, I handle nested parts in the beginning. Every path turned into absolute path and with string manipulation, checked if any nested paths exist. If so, I eliminated nested path.

There are 3 different storage element:

hfiles (int,list(str)): The map that maps file hashes to full paths of files that has same hash value.

hdirs (int,list(str)): The map that maps directory hashes to full paths of directories that has same hash value.

dirsizes (int,int): The map that maps directory hashes to sizes of directories.

Then I traverse every directory one by one, by traverse(dir) function in a recursive manner. Traverse(dir):

It traverses all sub-directories of given "dir" and fills the corresponding hash and size tables. It works in an iterative way. Traverses its sub-files and sub-directories and calculates hash values. When it has a file as child, calls hasher() function.

When it has a directory, calls itself. Iteration stops when it has no children as directory. When it is an empty directory, hashes empty string in case of -c, hashes name of the dir otherwise.

Finds hash of the directory by concatenating sub-files or directories’ hashes. In case of -n or -cn, adds name of the direction beginning.

It also calculates size of the directory as sum of the sizes of its sub-elements and stores in the size map.

Next I should talk about hasher(name) function which handles the hashing process of files. Hasher(name):

It calculates hash for a file and add it to corresponding storage element which is hfiles. Hashes according to the arguments "args" using sha256() as follows:

-n: hashes the name of the file

-c: hashes the content of the file via reading file as rb

-cn: hashes the concatenation of the name and the content.

If "-n", size is assigned to 0. It adds [hash : [full_path]] pair to the hfiles dictionary if doesn't exist. If calculated hash exist in the dictionary, appends full_path to the list of that hash value.

At the end of the code, when all dictionaries filled according to the files and directories, program eliminates that are not needed and created a printList dictionary to print. PrintList stores 2 types of maps in two different cases.

When -s case, printList maps size values to list of the full paths of that corresponding sized.

Otherwise, printList maps paths with same hashes, first path maps to rest of the paths list.

It is implemented in this way in order to sort paths easily with sort methods. In this manner, Key values can be sorted easily both cases when it is size or the first element. Every path list is also sorted in alphabetic order. After all sorting process, we have lists sorted by both alphabetic and sized.

Completion Status:

Project is done completely. There exist no bugs or missing parts encountered in test-cases prepared by me.

Notes:

- Program can handle -d -f cases together.
- The size of the empty directory accepted as 0 byte.
- Program takes -c -n -cn without problem but doesn’t accepts -c -n because Can Hoca explained in that way in the lectures.

Prepared by:

Hazar ÇAKIR

2017400093