BIS 420 PROGRAMMING FOR DATA SCIENCE

PRAJAKTA POHARE

CHAPTER 14 EXERCISE 14.4

ILLINOIS STATE UNIVERSITY

In a large collection of MP3 files, there may be more than one copy of the same song, stored in different directories or with different file names. The goal of this exercise is to search for duplicates.

1. Write a program that searches a directory and all of its subdirectories, recursively, and returns a list of complete paths for all files with a given suffix (like .mp3). Hint: os.path provides several useful functions for manipulating file and path names.

2. To recognize duplicates, you can use md5sum to compute a “checksum” for each files. If two files have the same checksum, they probably have the same contents.

3. To double-check, you can use the Unix command diff.

```
import os

import hashlib

import subprocess


def find_files_with_suffix(root, suffix=".mp3"):

    matches = []

    for dirpath, _, filenames in os.walk(root):

        for filename in filenames:

            if filename.lower().endswith(suffix.lower()):

                full_path = os.path.join(dirpath, filename)

                matches.append(full_path)

    return matches


def compute_md5(filename, block_size=65536):
    hash_md5 = hashlib.md5()
    try:
        with open(filename, "rb") as f:
            for chunk in iter(lambda: f.read(block_size), b""):
                hash_md5.update(chunk)
    except Exception as e:
        print(f"Error reading file {filename}: {e}")
        return None
    return hash_md5.hexdigest()

def find_duplicates(file_list):
    checksum_map = {}
    duplicates = []

    for filepath in file_list:
        checksum = compute_md5(filepath)
        if checksum is None:
            continue
        if checksum in checksum_map:
            duplicates.append((filepath, checksum_map[checksum]))
        else:
            checksum_map[checksum] = filepath

    return duplicates

def confirm_with_diff(file1, file2):
    try:
        result = subprocess.run(['diff', file1, file2], capture_output=True)
        return result.returncode == 0
    except Exception as e:
        print(f"Error running diff: {e}")
        return False


def main():
    root_dir = input("Enter the path to search for .mp3 files: ")
    all_mp3_files = find_files_with_suffix(root_dir, ".mp3")
    print(f"Found {len(all_mp3_files)} mp3 files.")

    duplicates = find_duplicates(all_mp3_files)

    if not duplicates:
        print("No duplicate files found.")
    else:
        print("\nPossible duplicates:")
        for file1, file2 in duplicates:
            identical = confirm_with_diff(file1, file2)
            status = "Identical" if identical else "Different"
            print(f"\n{file1}\n{file2}\n→ {status}")


if __name__ == "__main__":
    main()
```

```
import os
import hashlib
import subprocess
```

```python
def find_files_with_suffix(root, suffix=".mp3"):
    matches = []
    for dirpath, _, filenames in os.walk(root):
        for filename in filenames:
            if filename.lower().endswith(suffix.lower()):
                full_path = os.path.join(dirpath, filename)
                matches.append(full_path)
    return matches

def compute_md5(filename, block_size=65536):
    hash_md5 = hashlib.md5()
    try:
        with open(filename, "rb") as f:
            for chunk in iter(lambda: f.read(block_size), b""):
                hash_md5.update(chunk)
    except Exception as e:
        print(f"Error reading file {filename}: {e}")
        return None
    return hash_md5.hexdigest()

def find_duplicates(file_list):
    checksum_map = {}
    duplicates = []

    for filepath in file_list:
        checksum = compute_md5(filepath)
        if checksum is None:
            continue
        if checksum in checksum_map:
            duplicates.append((filepath, checksum_map[checksum]))
        else:
            checksum_map[checksum] = filepath

    return duplicates

def confirm_with_diff(file1, file2):
    try:
        result = subprocess.run(['diff', file1, file2], capture_output=True)
        return result.returncode == 0
    except Exception as e:
        print(f"Error running diff: {e}")
        return False

def main():
    root_dir = input("Enter the path to search for .mp3 files: ")
    all_mp3_files = find_files_with_suffix(root_dir, ".mp3")
    print(f"Found {len(all_mp3_files)} mp3 files.")
    duplicates = find_duplicates(all_mp3_files)

    if not duplicates:
        print("No duplicate files found.")
    else:
        print("\nPossible duplicates:")
        for file1, file2 in duplicates:
            identical = confirm_with_diff(file1, file2)
            status = "Identical" if identical else "Different"
            print(f"\n{file1}\n{file2}\n→ {status}")

if __name__ == "__main__":
    main()
```