Sohan Chatterjee

Lab 11 Complexity Analysis

SSW 315

14 November 2022

Complexity Analysis #4 - Duplicate File Finder

Method	**Worst Case**	**Storage Complexity**	**Justification for non-constant complexities**
fileFinder(String path)	n=path	O(n)	Given a directory, the method will recursively call itself until a file is found within the directory for the entire directory through a for-loop.
getMD5(String path)	none	O(1)	
readableFileSize(long size)	none	O(1)	
findDuplicates(String path)	n=path	$O(n^2)$	A previous method with a complexity of n is called and there are a few nested for-loops within this method but no more than two loops, so the complexities of all loops are n^2. The overall complexity of the method becomes $O(n+n^2+n^2+n^2)$ which can be simplified to $O(n^2)$.
main(String[] args)	n=path	$O(n^2)$	The main method of the class calls the findDuplicates() method with a given path.

```
//Sohan Chatterjee
//SSW 315 Duplicate File Finder
//November 02, 2022
import java.io.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.math.BigInteger;
import java.text.DecimalFormat;
```

```
public class DuplicateFileFinder {
    static ArrayList<File> allFiles = new ArrayList<File>();

    private static File fileFinder(String path) {
        File file = new File(path);
        if (file.isDirectory()) {
            File[] files = file.listFiles();
            for (File f : files) {
                fileFinder(f.getPath());
            }
        } else {
            allFiles.add(file);
        }
        return file;
    }

    private static String getMD5(String path) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            byte[] messageDigest = md.digest(path.getBytes());
            BigInteger no = new BigInteger(1, messageDigest);
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    private static String readableFileSize(long size) {
        if (size <= 0)
            return "";
        final String[] units = new String[] { "B", "kB", "MB", "GB", "TB"
};
        int digitGroups = (int) (Math.log10(size) / Math.log10(1024));
        return new DecimalFormat("#,##0.##").format(size / Math.pow(1024,
digitGroups)) + " " + units[digitGroups];
    }
    public static void findDuplicates(String path) throws IOException {
        fileFinder(path);
        boolean[] checked = new boolean[allFiles.size()];
        ArrayList<ArrayList<File>> duplicates = new ArrayList<>();

        for (int i = 0; i < allFiles.size() - 1; i++) {
            if (checked[i])
                continue;
            ArrayList<File> temp = new ArrayList<>();
            for (int j = i + 1; j < allFiles.size(); j++) {
                if ((allFiles.get(i).length() ==
allFiles.get(j).length())) {
                    temp.add(allFiles.get(j));
                    checked[j] = true;
                }
            }
            if (temp.size() > 0) {
                temp.add(allFiles.get(i));
                duplicates.add(temp);
            }
        }
        for (int i = 0; i < duplicates.size(); i++) {
            System.out.println("#   " + duplicates.get(i).size() + "    "
                    + readableFileSize(duplicates.get(i).get(0).length())
+ "   "
                    + getMD5(duplicates.get(i).get(0).getPath()));
            for (int j = 0; j < duplicates.get(i).size(); j++) {
                System.out.println(duplicates.get(i).get(j).getPath());
            }
            System.out.println();
        }

        File f = new File(path + ".log");
        if (!f.exists()) {
            try {
                f.createNewFile();
            } catch (IOException e) {
                System.out.println("could not create new log file");
            }
        }

        FileWriter fstream;
        try {
            fstream = new FileWriter(f, true);
            BufferedWriter out = new BufferedWriter(fstream);
            for (int i = 0; i < duplicates.size(); i++) {
                out.write("#   " + duplicates.get(i).size() + "    "
                        +
readableFileSize(duplicates.get(i).get(0).length()) + "   "
                        + getMD5(duplicates.get(i).get(0).getPath()));
                out.newLine();
                for (int j = 0; j < duplicates.get(i).size(); j++) {
                    out.write(duplicates.get(i).get(j).getPath());
                    out.newLine();
                }
                out.newLine();
            }
            out.close();
        } catch (IOException e) {
            System.out.println("could not write to the file");
        }
    }

    public static void main(String[] args) {
        try {
            findDuplicates("C:/Users/sohan/Documents/Stevens/2022 Fall/SSW
315/Duplicate File Finder/test images");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```