Plagiarism System

Yu Yu, Michael Reilly, Pengyu Su

Get the similarity of two source code

- Used Longest Common Subsequence (LCS) algorithm.

- LCS:

  The longest common subsequence problem is the problem of the finding the longest subsequence common to all sequences.

		C	B	C	A	B	C
	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	0	1	1	1	2	2
C	0	1	1	2	2	2	3
B	0	1	2	2	2	3	3
B	0	1	2	2	2	3	3

value

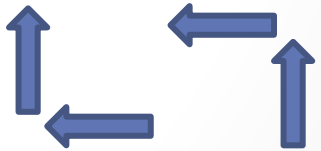
a	
	a+1

value

matc
h

	c
b	max(b or c)

not
match



		C	B	C	A	B	C
	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	0	1	1	1	2	2
C	0	1	1	2	2	2	3
B	0	1	2	2	2	3	3
B	0	1	2	2	2	3	3

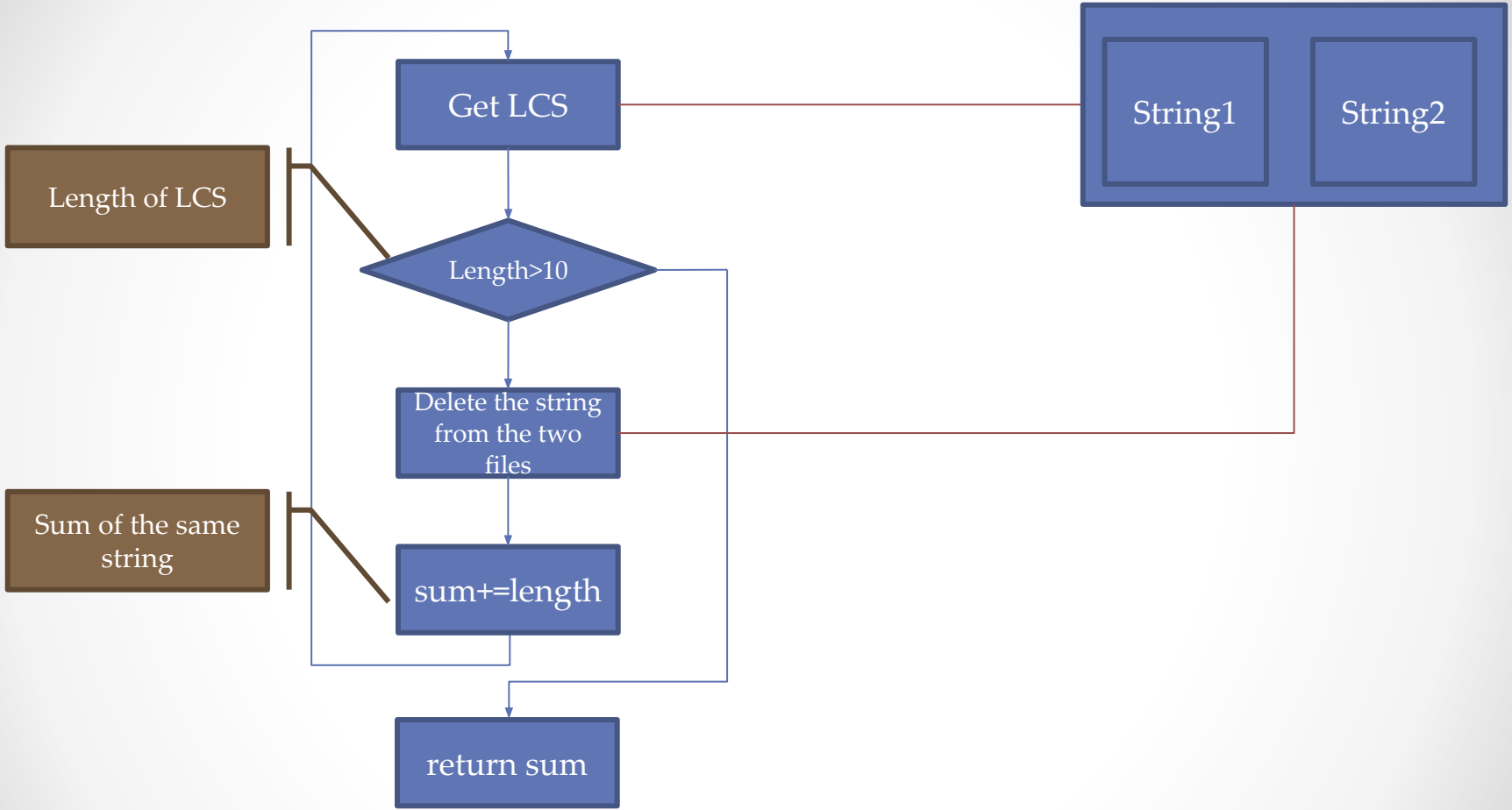
a	
	a+1

matc
h

	c
b	max(b or c)

not
match

LCS : ABC


Get LCS
String1
String2
Length of LCS
Length>10
Delete the string from the two files
Sum of the same string
sum+=length
return sum

Detect Plagiarism in Compiled File

1: Compile .java, .cpp, .py file

2: Compare .class, .obj, .pyc file using “Edit Distance Algorithm”

Compile .java File

```
JavaCompiler javaCompiler = ToolProvider.getSystemJavaCompiler();
JavaFileObject javaFileObject = new JavaStringObject(name,code);
CompilationTask task = javaCompiler.getTask(null, null, null,
        Arrays.asList("-d",direction), null, Arrays.asList(javaFileObject));
boolean success = task.call();
if(! success){
    System.out.println(name + " Compilation fail!");
}else{
    System.out.println(name + " Compilation success!");
    try {
        /*Automatically execute the .class file CompilationTask just made. Th
         * essential in plagiarism detection but here just add this function.
         */
        URL[] urls = new URL[]{
                new URL("file:/" + direction + "/")
        };
        URLClassLoader classLoader = new URLClassLoader(urls);
        Class class_ = classLoader.loadClass(name);
        Method  method = class_.getDeclaredMethod("main", String[].class);
        String[] args_ ={null};
        System.out.println("\nBelow is the output of " + name + ":");
        method.invoke(class_.newInstance(), args_);
        System.out.println(name + " output end!\n");
```

javax.tools.JavaCompiler
javax.tools.JavaCompiler.CompilationTask
javax.tools.JavaFileObject
javax.tools.ToolProvider

Compile .cpp File

```
public void compileSourceCode () {
    // TODO Auto-generated method stub
    String path = file.getPath().replaceFirst(file.getName(),"");
    String command = "cmd /c vcvarsall.bat&&" + path +
            "&&cl /EHsc " + file.getName();
    Process p;
    try {
        p = Runtime.getRuntime().exec(command);
        BufferedReader brTrue = new BufferedReader(new InputStreamRead
        BufferedReader brFalse = new BufferedReader(new InputStreamRea
        String line;
        while( (line = brTrue.readLine()) != null) {
            System.out.println(line);
        }
        if((line = brFalse.readLine()) != null){
            System.out.println("Error in Commond-Line!\n");
            System.out.println(line);
            while( (line = brFalse.readLine()) != null) {
                System.out.println(line);
            }
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

Runtime Class

Runtime.exec(command)

Command1: Vcvarsall.bat (Visual Studio 2015)

Command2: cl /EHsc xxx.cpp

Compile .py File

```
public void compileSourceCode() {
    // TODO Auto-generated method stub

    String path = file.getPath().replaceFirst(file.getName(),"");
    String command = "cmd /c " + path + "&&python -m py_compile " + file.getName();
    Process p;
    try {
        p = Runtime.getRuntime().exec(command);
        BufferedReader brTrue = new BufferedReader(new InputStreamReader(p.getInputS
        BufferedReader brFalse = new BufferedReader(new InputStreamReader(p.getError
        String line;
        while( (line = brTrue.readLine()) != null) {
            System.out.println(line);
        }
        if((line = brFalse.readLine()) != null){
            System.out.println("Error in Commond-Line!\n");
            System.out.println(line);
            while( (line = brFalse.readLine()) != null) {
                System.out.println(line);
            }
        }
```

Runtime Class
Runtime.exec(command)

Command:
Python -m py_compile xxx.py

Edit Distance Algorithm (Levenshtein Distance)

		i	v	a	n	1
	0	1	2	3	4	5
i	1					
v	2					
a	3					
n	4					
2	5					

Table[i][j]: Operation needed for change str1[0,i] into str[0,j]

		i	v	a	n	1
	0+t =0	**1+1 =2**	2	3	4	5
i	**1+1 =2**	0				
v	2					
a	3					
n	4					
2	5					

If (s1[i] == s2[j]) t = 0
Else t = 1

$$T(i,j) = \min\{\ T(i-1,j)+1\ ,\ T(i,j-1)+1\ ,\ T(i-1,j-1)+t\ \}$$

Edit Distance Algorithm (Levenshtein Distance)

		i	v	a	n	1
	0	1	2	3	4	5
i	1	0	1	2	3	4
v	2	1	0	1	2	3
a	3	2	1	0	1	2
n	4	3	2	1	0	1
2	5	4	3	2	1	1

Final Distance:

n1

Table[s1.length][s2.length]

Minimum single character edit operation (e.g. : deletion, insertion and substitution)

Syntax Assessment

- Removal of Keywords, added small characters to replace
- Removed Variables names and Primitives
  - Kept separated to assess if names were just copied
- Comments, spacings, etc. all removed
- All that's left is just a bunch of user-created strings

```
public class CodeCruncher {
                        //Looks for words and lines for parsing
                        private ArrayList<String> files;
                        private int overallScore;

                        //List of files
                        public CodeCruncher()
                        {
                                files = new ArrayList<String>();
                                overallScore = 0;
                        }
                        //Array List constructor with all the files
                        public CodeCruncher(ArrayList<String> files)
                        {
                                        this.files = files;
                                        overallScore = 0;
                        }
                        public int getOverallScore() {
                                return overallScore;
                        }

                        public double runOverallScore(File file1, File file2) throws IOException,
FileNotFoundException {
                                double score = 0;

                                //FileInputStream fis1 = null;
                                //BufferedReader reader1 = null;
                                LineParser lineparse1 = new LineParser();
                                File fileEdit1 = new
File(file1.getPath().replaceFirst(file1.getName(),file1.getName()+"Edit1"));

                                try (FileInputStream fis1  = new FileInputStream(file1);
                                BufferedReader reader1 =  new BufferedReader (new
InputStreamReader(fis1));)
                                {
                                        //fis1 = new FileInputStream(file1);
                                        //reader1 = new BufferedReader (new
InputStreamReader(fis1));

                                        FileOutputStream fos1 = new FileOutputStream(fileEdit1);

                                        byte[] writing = new byte[1];
```

pppcccCodeCruncherpppArrayList<String>filespppintoverallScorepppCodeCruncherfiles=newArrayList<String>overallScore=0pppCodeCruncher(ArrayList<String>files)this.files=filesoverallScore=0pppintgetOverallScorereturnoverallScorepppdoublerunOverallScore(Filefile1,Filefile2)throwsIOException,FileNotFoundExceptiondoublescore=0LineParserlineparse1=newLineParserFilefileEdit1=newFile(file1.getPath.replaceFirst(file1.getName,file1.getName+"Edit1"))try(FileInputStreamfis1=newFileInputStream(file1)BufferedReaderreader1=newBufferedReader(newInputStreamReader(fis1)))FileOutputStreamfos1=newFileOutputStream(fileEdit1)byte[]writing=newbyte[1]

Final Output

- All the Algorithms are summed together
  - Each is from a scale from 0 to 1
- If the sum is above a threshold
  - Requires manual investigation
- If below
  - Don't worry about it!

Future Improvement

1: Compiling .cpp file need a compiler(Visual Studio 2015 in this program which need to set “Environment Variable”), g++ is much convenient.

2: Compiling process will create .class file, .obj file and .pyc, we need a garbage cleaner.

3: More test data needed, so we can determine the parameter of plagiarism threshold.

4: Student Patterns, keeping track of whether their code changed stylistically between assignments

5: Checking if they credited the code to another student

6: Tracking variable that aren't just primitive types

THANK YOU

...

Q&A