1. **The source code of the file system traversal tool that you implemented in the Java language.**

From The TreeTraversal.java:

```
import java.io.File;
import java.util.*;

import com.beust.jcommander.JCommander;
public class TreeTraversal {

	long noOfFiles; //number of files variable
	long sizeOfFiles; //size of files variable
	long minimumSize; //minimum size of files variable
	long maximumSize; //maximum size of files variable
	long noOfDirectories; //number of directories variable
	long maxNoOfLevels; //max number of levels variable
	long currentLevel; //current level variable

	public void traverse(File file)
	{

		//This will print out all of the paths of the files of a root specified.
		//System.out.println(file.getAbsolutePath());

		//if it is a file
		if(file.isFile()){
			//if the file is a file
			//System.out.println("File: " + file.getAbsolutePath());
			//print out that it is a file and increase the number of files by 1
			noOfFiles ++;
			//increase the size of files by the length of each file in a root specified
			sizeOfFiles += file.length();
			//if the minimum size is zero, then the minimum size is the length of the
file
			if(minimumSize == 0) minimumSize = file.length();
```

```
                    //if the minimum is greater than the length of the file, then change the
minimum size file
                    if(minimumSize > file.length()) minimumSize = file.length();
                    //if the maximum is less than the length of a file, then change the
maximum size file.
                    if(maximumSize < file.length()) maximumSize = file.length();
                }
                //If it is a directory
                if (file.isDirectory()){
                    //if the file is a directory
                    //System.out.println("Directory: " + file.getAbsolutePath());
                    //show that it is a directory that is being going through

                    noOfDirectories++;
                    //increase the number of directories by 1
                    //array to go through the list of files
                    String listOfFiles[] = file.list();

                    if ((listOfFiles != null) && (currentLevel < maxNoOfLevels))
                        //if the list of files is not zero, go through the traverse
                    {
                        currentLevel++;


                        for (String filename : listOfFiles)
                            //for each file in the list of files
                        {
                            traverse(new File(file,filename));
                            //run the traverse tool
                        }
                        currentLevel--;
                    }
                }
            }
```

```
	public static void main(String[] args) {
		parameters param = new parameters();
		//call the parameters file
		new JCommander(param, args);
		//call the JCommander the parse the command line
		TreeTraversal Tree = new TreeTraversal();
		//call the TreeTraversal object

		File rootDirectory = new File(param.root);
		//call the root parameter
		if(param.levels == 0) {
			//if the level specified in the command line is 0
			Tree.maxNoOfLevels = Integer.MAX_VALUE;
			//the max number of levels is the maximum value for an integer
		}else{
			Tree.maxNoOfLevels = param.levels;
			//else the max number of levels is the levels parameter
		}

		long startTime = System.currentTimeMillis();
		//start time variable to show how long it takes to run the root

		Tree.traverse(rootDirectory);
		//traverse the root directory
		long timeSpend = System.currentTimeMillis() - startTime;
		//the time Spend is the current time minus the start time to get the time of how
long it takes to traverse the specific root.
		System.out.println("Root Directory: "+param.root);
		//print out the root parameter
		System.out.println("Level: "+param.levels);
		//print out the level parameter


		System.out.println("Number of files: " + Tree.noOfFiles);
		//print out the number of files that has been examined
```

```
                System.out.println("Number of directories: " + Tree.noOfDirectories);
                //print out the number of directories that has been examined
                System.out.println("Maximum file size: " + Tree.maximumSize + " bytes");
                //print out the max file size of all of the files examined
                System.out.println("Minimum file size: " + Tree.minimumSize + " bytes");
                //print out the minimum file size of all of the files examined
                System.out.println("Average file size: " + (float)(Tree.sizeOfFiles /
Tree.noOfFiles)    + " bytes");
                //print out the average file size of all the files examined as a float


                System.out.println("Time Spend: " + (float)(timeSpend/1000.00) + " seconds");
                //print out the time spend in seconds as a float

        }

}
```

From the parameters.java:

```
import com.beust.jcommander.Parameter;

public class parameters {

	@Parameter (names = {"-directory","-d"}, description = "Root directory for stats")
	String root;
//Call a specific root to go through to traverse
	@Parameter (names = {"-levels","-l"}, description = "Number of levels to traverse")
	Integer levels = 0;
//Call a specific level to go through a specific part of a computer
	@Parameter(names = "-debug", description = "Debug mode")


	private boolean debug = false;


}
```

2. **The output of the file system traversal tool when run on several small file system regions**

1)Here is the output from when I was running my tool on the Desktop:
Root Directory: /Users/Andreas/Desktop
Level: 0
Number of files: 30862
Number of directories: 4167
Maximum file size: 576716800 bytes
Minimum file size: 10 bytes
Average file size: 123276.0 bytes
Time Spend: 8.982 seconds

2) Here is the output from when I was running my tool on the Documents root:

Root Directory: /Users/Andreas/Documents
Level: 0
Number of files: 3837
Number of directories: 945
Maximum file size: 20325153 bytes
Minimum file size: 1 bytes
Average file size: 39648.0 bytes
Time Spend: 1.476 seconds

3) Here is the output from when I was running my tool on the Downloads:

Root Directory: /Users/Andreas/Downloads
Level: 0
Number of files: 202
Number of directories: 39
Maximum file size: 255649794 bytes
Minimum file size: 8 bytes
Average file size: 1990342.0 bytes
Time Spend: 0.086 seconds

4) Here is the output from when I was running the tool on my cs290 final project:

Root Directory: /Users/Andreas/Documents/College/2ndYear/1stSemester/cs290/finalproject/cs290f2013-fp-team3
Level: 0
Number of files: 233
Number of directories: 72
Maximum file size: 7383085 bytes
Minimum file size: 8 bytes
Average file size: 87694.0 bytes
Time Spend: 0.026 seconds

5) Here is the output from when I was running the tool on my workspace for my work in Eclipse

Root Directory: /Users/Andreas/Documents/workspace
Level: 0
Number of files: 461
Number of directories: 233
Maximum file size: 11277826 bytes
Minimum file size: 1 bytes
Average file size: 43253.0 bytes
Time Spend: 0.069 seconds