GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computing & Mathematics

CP2SD – Data Structures & Algorithms
Lab 1: Space and Time Complexity

A running programme exhibits both a space and a time complexity. Space complexity refers to the amount of memory consumed by an algorithm or application. In programming languages like C or C++, which are compiled into native machine code, memory complexity refers to the amount of Random Access Memory (RAM) that the algorithm consumes. In the Java programming language however, the runtime environment is the Java Virtual Machine (JVM), where memory consumption is reflected by the size of the object heap. Time complexity relates to the running time of an algorithm. The commonly accepted metric for approximating running time is big-O notation. In this practical, we will examine the space and time complexity of a simple application that searches a sorted array of words generated from a dictionary.

Do not use Eclipse for this lab. Compile and execute the source from a command line.

1. ***Download*** and extract the source code and dictionary from Moodle and copy the files to a local directory. Make sure you ***package the classes correctly***!
2. ***Compile*** the source and ***execute*** the example using the fully qualified package name: *java gmit.RunningTime*.
3. ***Uncomment*** each of the following methods and examine the running time: *isArrayOver100(), contains(), binarySearch()*.
4. ***Open*** the file *RunningTime.java* in a text editor. ***Add*** an instruction at the beginning of the *main()* method to cause the programme to wait for user input *(System.in.read())*. ***Uncomment*** the method *containsDuplicates()* and ***compile*** the Java source.
5. ***Execute*** the example using and instruct the JVM to allow remote monitoring of system resources: *java -Dcom.sun.management.jmxremote gmit.RunningTime*
6. When the programme stops for your input, open a new command prompt and start the ***JConsole*** monitor. ***Connect*** to the local JVM and switch to the heap monitor.
7. ***Continue*** with the *RunningTime* application by entering a number and pressing enter. You should now be able to monitor the space and time complexity of the *containsDuplicates()* method.
8. ***Estimate*** the amount of memory consumed by the methods in (3) using the following table:

Type	**Bytes**	**Type**	**Bytes**
boolean	1	byte[]	16 + n
byte	1	boolean[]	16 + n
char	2	char[]	16 + 2n
int	4	int[]	16 + 4n
float	4	double[]	16 + 8n
long	8	int[][]	4n2 + 20n + 16
double	8	double[][]	8n2 + 20n + 16
Object Ref	4	String	40 + 2n