Experimental Project: Longest Common Subsequence - Complete Implementation

Aishwary Pramanik

***Abstract*—An algorithmic approach to find Longest Common Sub sequence of two input strings and observe the performance patterns, considering various data inputs in different scenarios.**

I. Introduction

THIS approach comprises of different modules to observe the performance trends while finding the Longest Common Sub sequence of two input Strings. Firstly, two random strings are generated consisting of either binary numbers or alphabets. Then implementing the respective algorithms with the help of a profiler and analyzing the results in terms of cpu measurements and minimizing the user interface.

II. Configuration

A. System Configuration

Operating System	Windows 10 Home
Processor	i7-4710HQ CPU @ 2.50GHz
RAM	16.0GB
System Type	64-bit Operating System, x64-based processor
Programming Language	Java
IDE	Eclipse Java EE IDE, Mars.1 Release (4.5.1)

B. Profiler

The profiler being used for this approach is YourKitJava-Trial. It works to as a plugin in Eclipse and runs in background to generate CPU measurements. Features provided are:
1.Platform independent
2.Local and remote profiling
3.Controllable overhead
4.Easy to use
5.Developer Friendly
6.CPU profiling
7.Memory Profiling
8.Threads, monitors, exceptions

C. Data Set

There are two type of data sets being used during the implementation stage viz.
1.Binary (0 or 1)
2.Alphabets (A to Z)

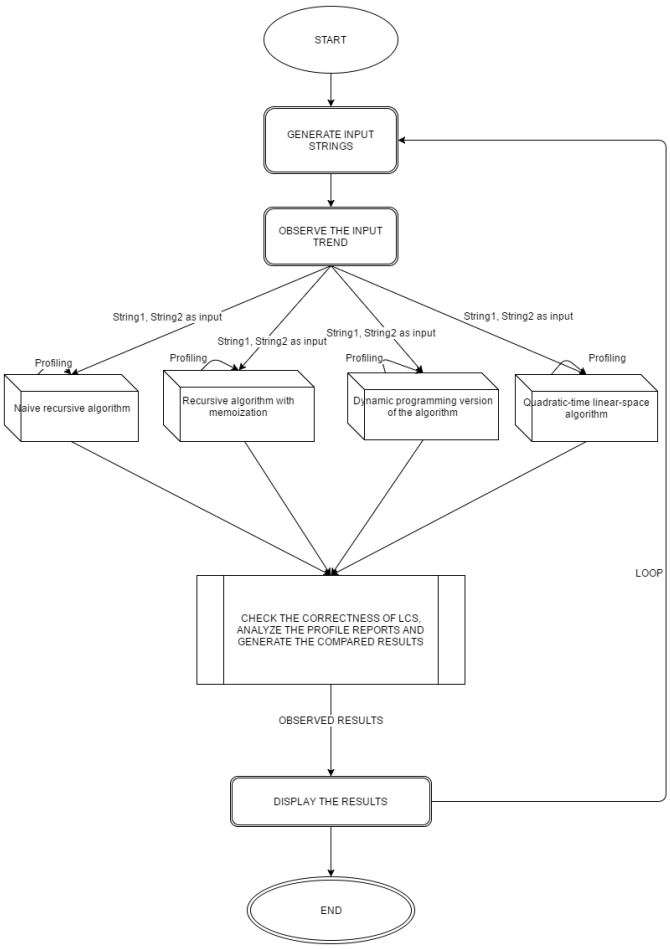


Fig. 1. Step by Step approach for complete process

III. Flowchart

IV. Classes

A. ExecuteAlgo

This Class generates two random input strings and passes them to the respective algorithmic class object methods(Eg. dynamicLCS.findLCS(String1 str1,String2 str2)). This class consists of a main method from where the overall execution starts and a comparer method which compares all the algorithms.

Advisor: Professor Stanisaw P. Radziszowski

B. dynamicLCS

This Class accepts two random input strings, finds the LCS with dynamic programming approach, perform profiling and returns profile report as well as the LCS.

C. naiveLCS

This Class accepts two random input strings, finds the LCS with Naive recursive algorithm approach, perform profiling and returns profile report as well as the LCS.

D. recMemoizedLCS

This Class accepts two random input strings, finds the LCS with Recursive algorithm with memoization approach, perform profiling and returns profile report as well as the LCS.

E. lsqtLCS

This Class accepts two random input strings, finds the LCS with Quadratic-time linear-space algorithm approach, perform profiling and returns profile report as well as the LCS.

V. High Level Script

A. Generates the most relevant output where the CPU time exceeds 10seconds limit.

B. Compares all the algorithms and generates resulting comparative graphs

C. The high level script with the help of profiler, generates output in following format:

CPU time for execution	Length of String1	Length of String2	Length of LCS

VI. Observations

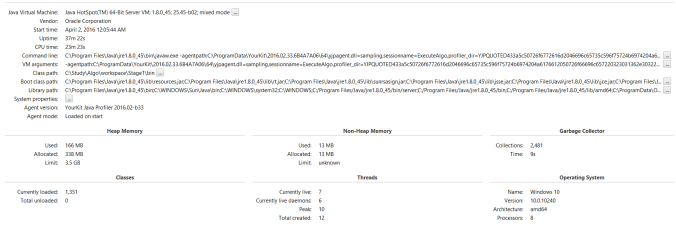


Fig. 2. Summary of Observations

All the observations are attached behind this report.

VII. Previous Version

A. Algorithm(s) Implemented

1. Dynamic programming version of the algorithm

B. Algorithm(s) to be Implemented

1. Naive recursive algorithm 2. Recursive algorithm with Memoization 3. Quadratic-time linear-space algorithm

VIII. Current Version

In current version of this project, I have added execution of remaining three algorithms with the help of following methods:

1. naiveLCS
2. recMemoizedLCS
3. lsqtLCS

IX. Project Status

Implemented following algorithms to find out how accurate are the theoretical estimates of complexity when compared to practical execution times:

1. Dynamic programming version of the algorithm
2. Naive recursive algorithm
3. Recursive algorithm with memoization
4. Quadratic-time linear-space algorithm

X. Future Scope

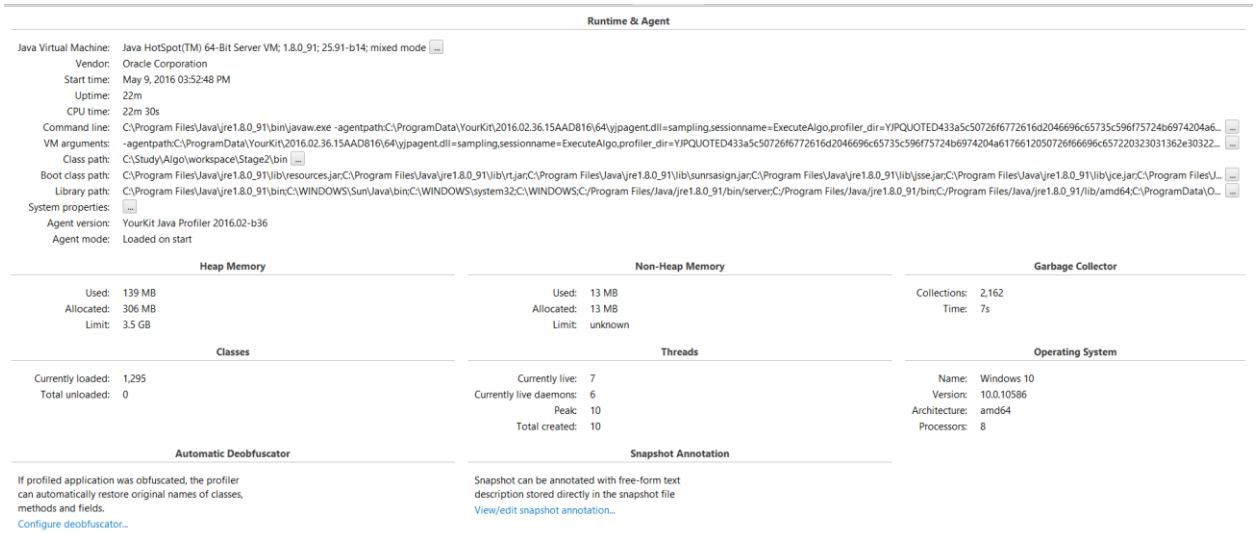
Reuse the basic structure of the implemented algorithms, for solving problems from different domains.

References

[1] IEEE Reference. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5768323&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5768323
[2] HirschbergD.S.: Algorithms for the Longest Common Subsequence Problem J.ACM ,Vol.24, No.4, October1977, PP.664-675. http://dl.acm.org/citation.cfm?doid=322033.322044
[3] Introduction to Algorithms 3rd Edition - Cormen, Leiserson, Rivest, Stein
[4] YourKit-Trial Profiler https://www.yourkit.com/java/profiler/features/

EXECUTION SUMMARY

Dynamic:



Runtime & Agent

Java Virtual Machine:	Java HotSpot(TM) 64-Bit Server VM; 1.8.0_91; 25.91-b14; mixed mode ...
Vendor:	Oracle Corporation
Start time:	May 9, 2016 03:52:48 PM
Uptime:	22m
CPU time:	22m 30s
Command line:	C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe -agentpath:C:\ProgramData\YourKit\2016.02.36.15AAD816\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a6... ...
VM arguments:	-agentpath:C:\ProgramData\YourKit\2016.02.36.15AAD816\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a6176612050726f66696c6572203231362e30322... ...
Class path:	C:\Study\Algo\workspace\Stage2\bin ...
Boot class path:	C:\Program Files\Java\jre1.8.0_91\lib\resources.jar;C:\Program Files\Java\jre1.8.0_91\lib\rt.jar;C:\Program Files\Java\jre1.8.0_91\lib\sunrsasign.jar;C:\Program Files\Java\jre1.8.0_91\lib\jsse.jar;C:\Program Files\Java\jre1.8.0_91\lib\jce.jar;C:\Program Files\J... ...
Library path:	C:\Program Files\Java\jre1.8.0_91\bin;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32;C:\WINDOWS;C:/Program Files/Java/jre1.8.0_91/bin/server;C:/Program Files/Java/jre1.8.0_91/bin;C:/Program Files/Java/jre1.8.0_91/lib/amd64;C:\ProgramData\O... ...
System properties:	...
Agent version:	YourKit Java Profiler 2016.02-b36
Agent mode:	Loaded on start

Heap Memory

Used:	139 MB
Allocated:	306 MB
Limit:	3.5 GB

Non-Heap Memory

Used:	13 MB
Allocated:	13 MB
Limit:	unknown

Garbage Collector

Collections:	2,162
Time:	7s

Classes

Currently loaded:	1,295
Total unloaded:	0

Threads

Currently live:	7
Currently live daemons:	6
Peak:	10
Total created:	10

Operating System

Name:	Windows 10
Version:	10.0.10586
Architecture:	amd64
Processors:	8

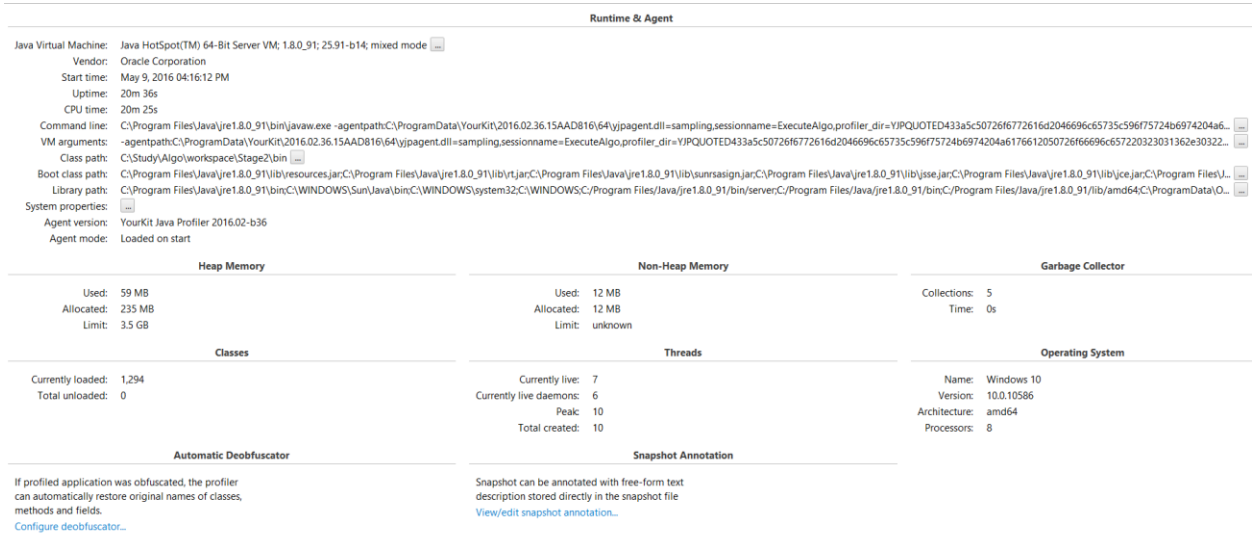
Automatic Deobfuscator

If profiled application was obfuscated, the profiler can automatically restore original names of classes, methods and fields.
Configure deobfuscator...

Snapshot Annotation

Snapshot can be annotated with free-form text description stored directly in the snapshot file
View/edit snapshot annotation...

Linear Space Quadratic Time:



Runtime & Agent

Java Virtual Machine:	Java HotSpot(TM) 64-Bit Server VM; 1.8.0_91; 25.91-b14; mixed mode ...
Vendor:	Oracle Corporation
Start time:	May 9, 2016 04:16:12 PM
Uptime:	20m 36s
CPU time:	20m 25s
Command line:	C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe -agentpath:C:\ProgramData\YourKit\2016.02.36.15AAD816\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a6... ...
VM arguments:	-agentpath:C:\ProgramData\YourKit\2016.02.36.15AAD816\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a6176612050726f66696c6572203231362e30322... ...
Class path:	C:\Study\Algo\workspace\Stage2\bin ...
Boot class path:	C:\Program Files\Java\jre1.8.0_91\lib\resources.jar;C:\Program Files\Java\jre1.8.0_91\lib\rt.jar;C:\Program Files\Java\jre1.8.0_91\lib\sunrsasign.jar;C:\Program Files\Java\jre1.8.0_91\lib\jsse.jar;C:\Program Files\Java\jre1.8.0_91\lib\jce.jar;C:\Program Files\J... ...
Library path:	C:\Program Files\Java\jre1.8.0_91\bin;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32;C:\WINDOWS;C:/Program Files/Java/jre1.8.0_91/bin/server;C:/Program Files/Java/jre1.8.0_91/bin;C:/Program Files/Java/jre1.8.0_91/lib/amd64;C:\ProgramData\O... ...
System properties:	...
Agent version:	YourKit Java Profiler 2016.02-b36
Agent mode:	Loaded on start

Heap Memory

Used:	59 MB
Allocated:	235 MB
Limit:	3.5 GB

Non-Heap Memory

Used:	12 MB
Allocated:	12 MB
Limit:	unknown

Garbage Collector

Collections:	5
Time:	0s

Classes

Currently loaded:	1,294
Total unloaded:	0

Threads

Currently live:	7
Currently live daemons:	6
Peak:	10
Total created:	10

Operating System

Name:	Windows 10
Version:	10.0.10586
Architecture:	amd64
Processors:	8

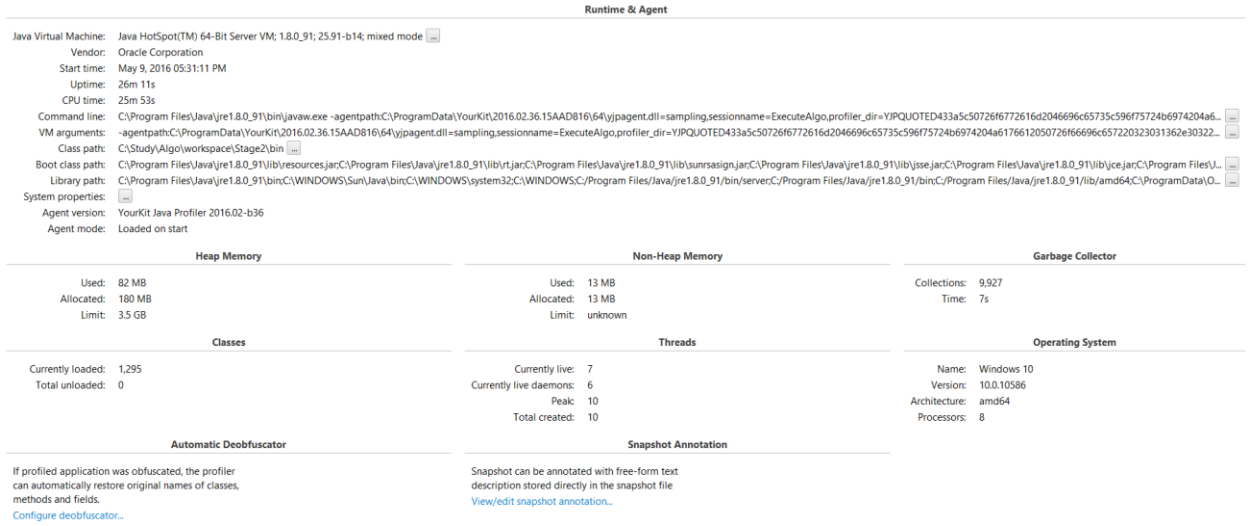
Automatic Deobfuscator

If profiled application was obfuscated, the profiler can automatically restore original names of classes, methods and fields.
Configure deobfuscator...

Snapshot Annotation

Snapshot can be annotated with free-form text description stored directly in the snapshot file
View/edit snapshot annotation...

Naïve:



Runtime & Agent

Java Virtual Machine:	Java HotSpot(TM) 64-Bit Server VM; 1.8.0_91; 25.91-b14; mixed mode ...
Vendor:	Oracle Corporation
Start time:	May 9, 2016 05:31:11 PM
Uptime:	26m 11s
CPU time:	25m 53s
Command line:	C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe -agentpath:C:\ProgramData\YourKit\2016.02.36.15AAD816\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a6... ...
VM arguments:	-agentpath:C:\ProgramData\YourKit\2016.02.36.15AAD816\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a617661205072f66696c657220323031362e30322... ...
Class path:	C:\Study\Algo\workspace\Stage2\bin ...
Boot class path:	C:\Program Files\Java\jre1.8.0_91\lib\resources.jar;C:\Program Files\Java\jre1.8.0_91\lib\rt.jar;C:\Program Files\Java\jre1.8.0_91\lib\sunrsasign.jar;C:\Program Files\Java\jre1.8.0_91\lib\jsse.jar;C:\Program Files\Java\jre1.8.0_91\lib\jce.jar;C:\Program Files\J... ...
Library path:	C:\Program Files\Java\jre1.8.0_91\bin;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32;C:\WINDOWS;C:/Program Files/Java/jre1.8.0_91/bin/server;C:/Program Files/Java/jre1.8.0_91/bin;C:/Program Files/Java/jre1.8.0_91/lib/amd64;C:\ProgramData\O... ...
System properties:	...
Agent version:	YourKit Java Profiler 2016.02-b36
Agent mode:	Loaded on start

Heap Memory

Used:	82 MB
Allocated:	180 MB
Limit:	3.5 GB

Non-Heap Memory

Used:	13 MB
Allocated:	13 MB
Limit:	unknown

Garbage Collector

Collections:	9,927
Time:	7s

Classes

Currently loaded:	1,295
Total unloaded:	0

Threads

Currently live:	7
Currently live daemons:	6
Peak:	10
Total created:	10

Operating System

Name:	Windows 10
Version:	10.0.10586
Architecture:	amd64
Processors:	8

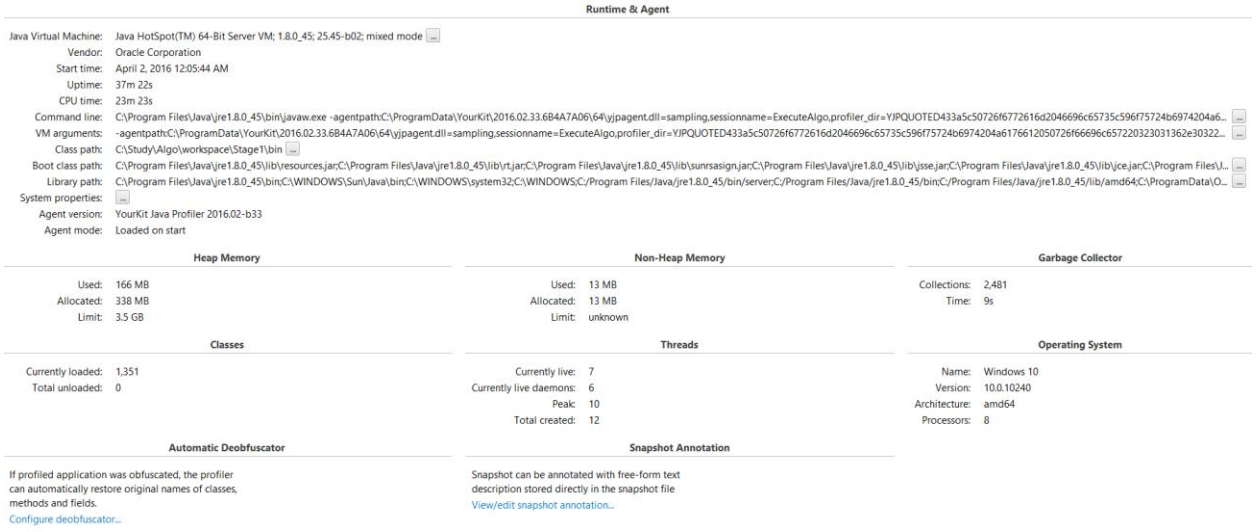
Automatic Deobfuscator

If profiled application was obfuscated, the profiler can automatically restore original names of classes, methods and fields.
Configure deobfuscator...

Snapshot Annotation

Snapshot can be annotated with free-form text description stored directly in the snapshot file
View/edit snapshot annotation...

Recursion with Memoization:



Runtime & Agent

Java Virtual Machine:	Java HotSpot(TM) 64-Bit Server VM; 1.8.0_45; 25.45-b02; mixed mode ...
Vendor:	Oracle Corporation
Start time:	April 2, 2016 12:05:44 AM
Uptime:	37m 22s
CPU time:	23m 23s
Command line:	C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe -agentpath:C:\ProgramData\YourKit\2016.02.33.6B4A7A06\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a6... ...
VM arguments:	-agentpath:C:\ProgramData\YourKit\2016.02.33.6B4A7A06\64\yjpagent.dll=sampling,sessionname=ExecuteAlgo,profiler_dir=YJPQUOTED433a5c50726f6772616d2046696c65735c596f75724b6974204a617661205072f66696c657220323031362e30322... ...
Class path:	C:\Study\Algo\workspace\Stage1\bin ...
Boot class path:	C:\Program Files\Java\jre1.8.0_45\lib\resources.jar;C:\Program Files\Java\jre1.8.0_45\lib\rt.jar;C:\Program Files\Java\jre1.8.0_45\lib\sunrsasign.jar;C:\Program Files\Java\jre1.8.0_45\lib\jsse.jar;C:\Program Files\Java\jre1.8.0_45\lib\jce.jar;C:\Program Files\J... ...
Library path:	C:\Program Files\Java\jre1.8.0_45\bin;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32;C:\WINDOWS;C:/Program Files/Java/jre1.8.0_45/bin/server;C:/Program Files/Java/jre1.8.0_45/bin;C:/Program Files/Java/jre1.8.0_45/lib/amd64;C:\ProgramData\O... ...
System properties:	...
Agent version:	YourKit Java Profiler 2016.02-b33
Agent mode:	Loaded on start

Heap Memory

Used:	166 MB
Allocated:	338 MB
Limit:	3.5 GB

Non-Heap Memory

Used:	13 MB
Allocated:	13 MB
Limit:	unknown

Garbage Collector

Collections:	2,481
Time:	9s

Classes

Currently loaded:	1,351
Total unloaded:	0

Threads

Currently live:	7
Currently live daemons:	6
Peak:	10
Total created:	12

Operating System

Name:	Windows 10
Version:	10.0.10240
Architecture:	amd64
Processors:	8

Automatic Deobfuscator

If profiled application was obfuscated, the profiler can automatically restore original names of classes, methods and fields.
Configure deobfuscator...

Snapshot Annotation

Snapshot can be annotated with free-form text description stored directly in the snapshot file
View/edit snapshot annotation...

Dynamic Vs Linear Space Quadratic Time

Call Tree	Time Diff (ms)	Old Time (ms)	New Time (ms)
<All threads>	-76,937	1,291,406	1,214,468
ExecuteAlgo.main(String[])	-76,937	1,291,406	1,214,468
ExecuteAlgo.java:126 **lsqtLCS.findLCS**(StringBuffer, StringBuffer)	+1,199,812	0	1,199,812
lsqtLCS.java:10	+474,390	0	474,390
lsqtLCS.java:11	+469,968	0	469,968
lsqtLCS.java:11 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	+134,562	0	134,562
lsqtLCS.java:55 java.lang.**StringBuffer.append**(char)	+134,515	0	134,515
lsqtLCS.java:47	+31	0	31
lsqtLCS.java:43	+15	0	15
lsqtLCS.java:10 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	+120,890	0	120,890
lsqtLCS.java:55 java.lang.**StringBuffer.append**(char)	+120,828	0	120,828
lsqtLCS.java:47	+62	0	62
ExecuteAlgo.java:130 java.io.**PrintStream.println**(String)	+14,296	156	14,453
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	+62	31	93
ExecuteAlgo.java:126 java.lang.**StringBuffer.<init>**(CharSequence)	+46	0	46
ExecuteAlgo.java:98	+31	0	31
ExecuteAlgo.java:57 java.util.**Scanner.<clinit>**()	+15	0	15
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	0	15	15
ExecuteAlgo.java:117 **DynamicLCS.findLCS**(StringBuffer, StringBuffer)	-1,291,203	1,291,203	0
DynamicLCS.java:46 java.lang.**StringBuffer.reverse**()	-15	15	0
DynamicLCS.java:30	-31	31	0
DynamicLCS.java:32	-78	78	0
DynamicLCS.java:45 java.lang.**StringBuffer.append**(char)	-593	593	0
DynamicLCS.java:23	-2,531	2,531	0
DynamicLCS.java:40	-1,287,953	1,287,953	0

Dynamic Vs Naïve

Call Tree	Time Diff (ms)	Old Time (ms)	New Time (ms)
<All threads>	-1,283,390	1,291,406	8,015
ExecuteAlgo.main(String[])	-1,283,390	1,291,406	8,015
ExecuteAlgo.java:116 java.lang.**StringBuffer.<init>**(CharSequence)	+3,484	0	3,484
ExecuteAlgo.java:116 **NaiveLCS.findLCS**(StringBuffer, StringBuffer)	+2,484	0	2,484
NaiveLCS.java:39	+640	0	640
NaiveLCS.java:36	+640	0	640
NaiveLCS.java:17	+578	0	578
NaiveLCS.java:20	+515	0	515
NaiveLCS.java:24 java.lang.**StringBuffer.append**(char)	+62	0	62
NaiveLCS.java:56	+31	0	31
NaiveLCS.java:41	+15	0	15
ExecuteAlgo.java:103	+1,125	0	1,125
ExecuteAlgo.java:126 java.io.**PrintStream.println**(String)	+609	0	609
ExecuteAlgo.java:105	+109	0	109
ExecuteAlgo.java:54 java.util.**Scanner.<init>**(InputStream)	+46	0	46
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(long)	+31	0	31
ExecuteAlgo.java:93	+31	0	31
ExecuteAlgo.java:94	+31	0	31
ExecuteAlgo.java:101	+31	0	31
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(int)	+15	0	15
ExecuteAlgo.java:124 java.lang.**System.currentTimeMillis**()	+15	0	15
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	-15	15	0
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	-31	31	0
ExecuteAlgo.java:130 java.io.**PrintStream.println**(String)	-156	156	0
ExecuteAlgo.java:117 **DynamicLCS.findLCS**(StringBuffer, StringBuffer)	-1,291,203	1,291,203	0
DynamicLCS.java:46 java.lang.**StringBuffer.reverse**()	-15	15	0
DynamicLCS.java:30	-31	31	0
DynamicLCS.java:32	-78	78	0
DynamicLCS.java:45 java.lang.**StringBuffer.append**(char)	-593	593	0
DynamicLCS.java:23	-2,531	2,531	0
DynamicLCS.java:40	-1,287,953	1,287,953	0

Dynamic Vs Recursion with Memoization

Call Tree	Time Diff (ms)	Old Time (ms)	New Time (ms)
<All threads>	-1,285,984	1,291,406	5,421
sun.launcher.**LauncherHelper.checkAndLoadMain**(boolean, int, String)	+31	0	31
ExecuteAlgo.main(String[])	-1,286,015	1,291,406	5,390
ExecuteAlgo.java:123 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+5,296	0	5,296
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	+31	31	62
ExecuteAlgo.java:57 java.util.**Scanner.<clinit>**()	+15	0	15
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	0	15	15
ExecuteAlgo.java:130 java.io.**PrintStream.println**(String)	-156	156	0
ExecuteAlgo.java:117 **DynamicLCS.findLCS**(StringBuffer, StringBuffer)	-1,291,203	1,291,203	0
DynamicLCS.java:46 java.lang.**StringBuffer.reverse**()	-15	15	0
DynamicLCS.java:30	-31	31	0
DynamicLCS.java:32	-78	78	0
DynamicLCS.java:45 java.lang.**StringBuffer.append**(char)	-593	593	0
DynamicLCS.java:23	-2,531	2,531	0
DynamicLCS.java:40	-1,287,953	1,287,953	0

Linear Space Quadratic Time Vs Naïve

Call Tree	Time Diff (ms)	Old Time (ms)	New Time (ms)
<All threads>	-1,206,453	1,214,468	8,015
ExecuteAlgo.main(String[])	-1,206,453	1,214,468	8,015
ExecuteAlgo.java:116 java.lang.**StringBuffer.<init>**(CharSequence)	+3,484	0	3,484
ExecuteAlgo.java:116 **NaiveLCS.findLCS**(StringBuffer, StringBuffer)	+2,484	0	2,484
NaiveLCS.java:39	+640	0	640
NaiveLCS.java:36	+640	0	640
NaiveLCS.java:17	+578	0	578
NaiveLCS.java:20	+515	0	515
NaiveLCS.java:24 java.lang.**StringBuffer.append**(char)	+62	0	62
NaiveLCS.java:56	+31	0	31
NaiveLCS.java:41	+15	0	15
ExecuteAlgo.java:103	+1,125	0	1,125
ExecuteAlgo.java:126 java.io.**PrintStream.println**(String)	+609	0	609
ExecuteAlgo.java:105	+109	0	109
ExecuteAlgo.java:54 java.util.**Scanner.<init>**(InputStream)	+46	0	46
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(long)	+31	0	31
ExecuteAlgo.java:93	+31	0	31
ExecuteAlgo.java:94	+31	0	31
ExecuteAlgo.java:101	+31	0	31
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(int)	+15	0	15
ExecuteAlgo.java:124 java.lang.**System.currentTimeMillis**()	+15	0	15
ExecuteAlgo.java:57 java.util.**Scanner.<clinit>**()	-15	15	0
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	-15	15	0
ExecuteAlgo.java:98	-31	31	0
ExecuteAlgo.java:126 java.lang.**StringBuffer.<init>**(CharSequence)	-46	46	0
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	-93	93	0
ExecuteAlgo.java:130 java.io.**PrintStream.println**(String)	-14,453	14,453	0
ExecuteAlgo.java:126 **lsqtLCS.findLCS**(StringBuffer, StringBuffer)	-1,199,812	1,199,812	0
lsqtLCS.java:10 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	-120,890	120,890	0
lsqtLCS.java:11 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	-134,562	134,562	0
lsqtLCS.java:11	-469,968	469,968	0
lsqtLCS.java:10	-474,390	474,390	0

Linear Space Quadratic Time Vs Recursion with Memoization

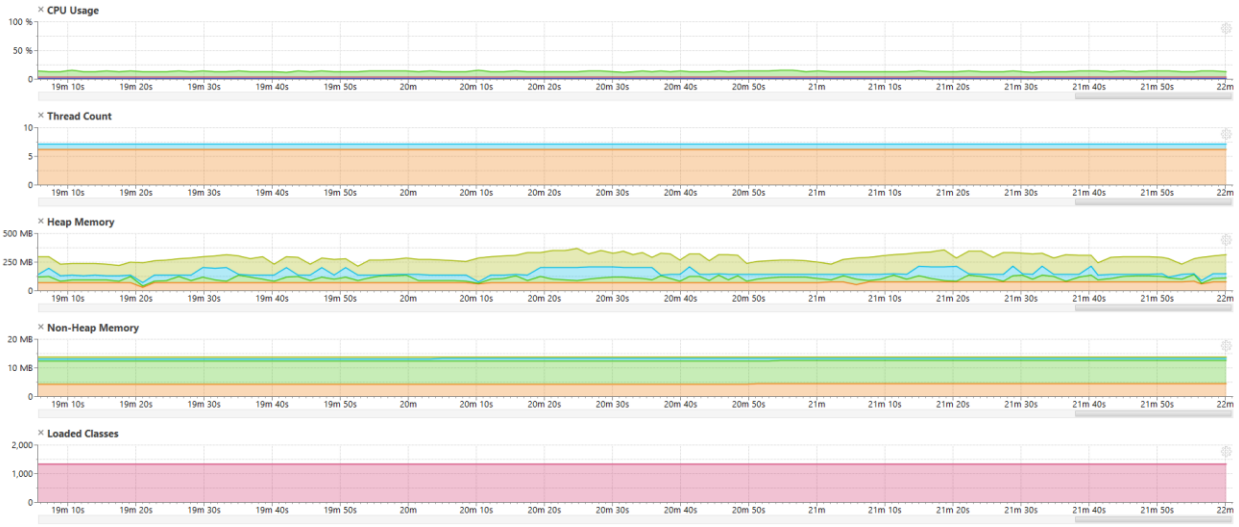
Call Tree	Time Diff (ms)	Old Time (ms)	New Time (ms)
<All threads>	-1,209,046	1,214,468	5,421
sun.launcher.**LauncherHelper.checkAndLoadMain**(boolean, int, String)	+31	0	31
ExecuteAlgo.main(String[])	-1,209,078	1,214,468	5,390
ExecuteAlgo.java:123 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+5,296	0	5,296
RecursionMemo.java:31 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+2,765	0	2,765
RecursionMemo.java:36 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+1,562	0	1,562
RecursionMemo.java:31 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+828	0	828
RecursionMemo.java:36 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+734	0	734
RecursionMemo.java:31 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+1,203	0	1,203
RecursionMemo.java:36 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+609	0	609
RecursionMemo.java:31 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+593	0	593
RecursionMemo.java:36 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	+2,531	0	2,531
ExecuteAlgo.java:57 java.util.**Scanner.<clinit>**()	0	15	15
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	0	15	15
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	-31	93	62
ExecuteAlgo.java:98	-31	31	0
ExecuteAlgo.java:126 java.lang.**StringBuffer.<init>**(CharSequence)	-46	46	0
ExecuteAlgo.java:130 java.io.**PrintStream.println**(String)	-14,453	14,453	0
ExecuteAlgo.java:126 **lsqtLCS.findLCS**(StringBuffer, StringBuffer)	-1,199,812	1,199,812	0
lsqtLCS.java:10 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	-120,890	120,890	0
lsqtLCS.java:47	-62	62	0
lsqtLCS.java:55 java.lang.**StringBuffer.append**(char)	-120,828	120,828	0
lsqtLCS.java:11 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	-134,562	134,562	0
lsqtLCS.java:43	-15	15	0
lsqtLCS.java:47	-31	31	0
lsqtLCS.java:55 java.lang.**StringBuffer.append**(char)	-134,515	134,515	0
lsqtLCS.java:11	-469,968	469,968	0
lsqtLCS.java:10	-474,390	474,390	0

Naïve Vs Recursion with Memoization

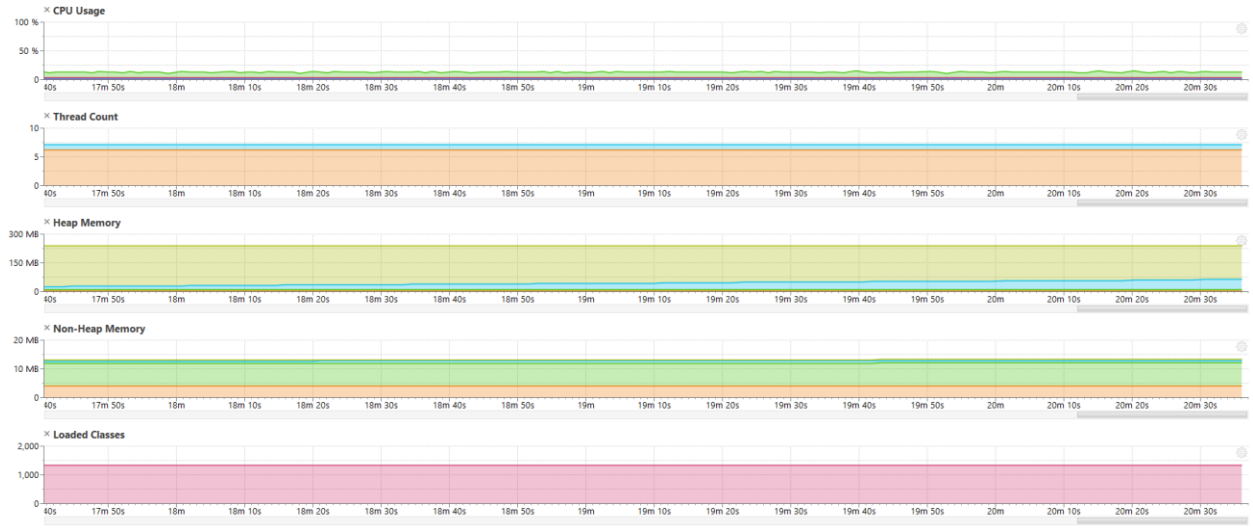
Call Tree	Time Diff (ms)	Old Time (ms)	New Time (ms)
<All threads>	+2,593	5,421	8,015
ExecuteAlgo.main(String[])	+2,625	5,390	8,015
ExecuteAlgo.java:116 java.lang.**StringBuffer.<init>**(CharSequence)	+3,484	0	3,484
ExecuteAlgo.java:116 **NaiveLCS.findLCS**(StringBuffer, StringBuffer)	+2,484	0	2,484
NaiveLCS.java:39	+640	0	640
NaiveLCS.java:36	+640	0	640
NaiveLCS.java:17	+578	0	578
NaiveLCS.java:20	+515	0	515
NaiveLCS.java:24 java.lang.**StringBuffer.append**(char)	+62	0	62
NaiveLCS.java:56	+31	0	31
NaiveLCS.java:41	+15	0	15
ExecuteAlgo.java:103	+1,125	0	1,125
ExecuteAlgo.java:126 java.io.**PrintStream.println**(String)	+609	0	609
ExecuteAlgo.java:105	+109	0	109
ExecuteAlgo.java:54 java.util.**Scanner.<init>**(InputStream)	+46	0	46
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(long)	+31	0	31
ExecuteAlgo.java:93	+31	0	31
ExecuteAlgo.java:94	+31	0	31
ExecuteAlgo.java:101	+31	0	31
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(int)	+15	0	15
ExecuteAlgo.java:124 java.lang.**System.currentTimeMillis**()	+15	0	15
ExecuteAlgo.java:57 java.util.**Scanner.<clinit>**()	-15	15	0
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	-15	15	0
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	-62	62	0
ExecuteAlgo.java:123 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	-5,296	5,296	0
RecursionMemo.java:36 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	-2,531	2,531	0
RecursionMemo.java:36 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	-1,171	1,171	0
RecursionMemo.java:31 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	-1,359	1,359	0
RecursionMemo.java:31 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	-2,765	2,765	0
sun.launcher.**LauncherHelper.checkAndLoadMain**(boolean, int, String)	-31	31	0

PERFORMANCE CHARTS

Dynamic:

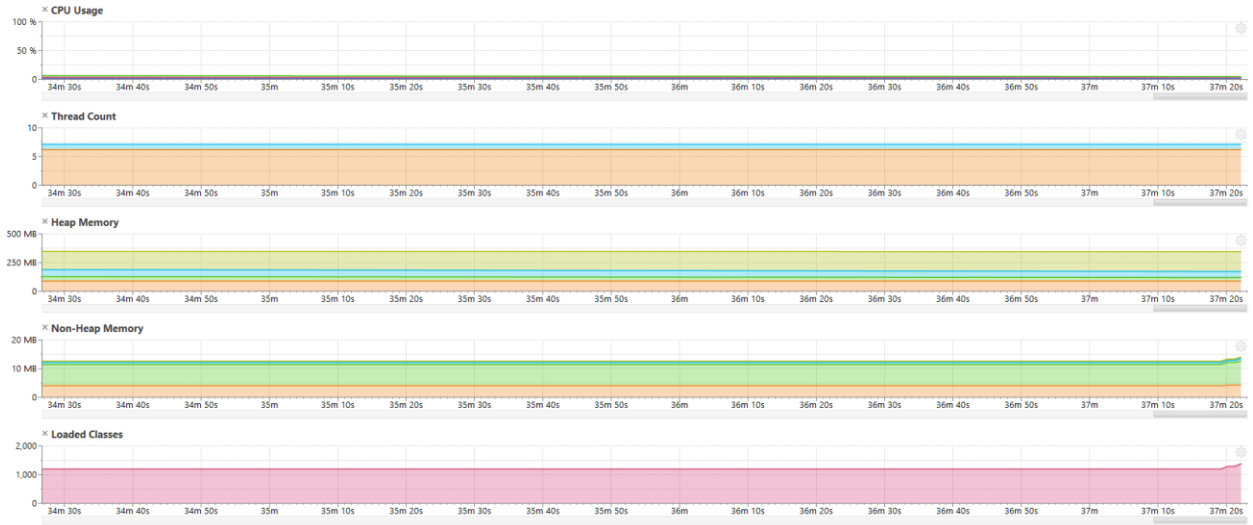


Linear Space Quadratic Time:



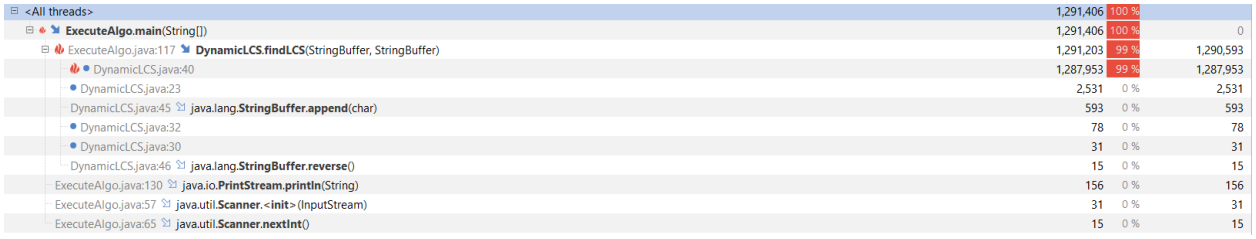
Naïve:

Recursion with Memoization:



THREAD EXECUTION

Dynamic:



<All threads>	1,291,406	100 %	
ExecuteAlgo.main(String[])	1,291,406	100 %	0
ExecuteAlgo.java:117 **DynamicLCS.findLCS**(StringBuffer, StringBuffer)	1,291,203	99 %	1,290,593
DynamicLCS.java:40	1,287,953	99 %	1,287,953
DynamicLCS.java:23	2,531	0 %	2,531
DynamicLCS.java:45 java.lang.**StringBuffer.append**(char)	593	0 %	593
DynamicLCS.java:32	78	0 %	78
DynamicLCS.java:30	31	0 %	31
DynamicLCS.java:46 java.lang.**StringBuffer.reverse**()	15	0 %	15
ExecuteAlgo.java:130 java.io.**PrintStream.println**(String)	156	0 %	156
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	31	0 %	31
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	15	0 %	15

Linear Space Quadratic Time:

<All threads>	1,214,468	100 %	
ExecuteAlgo.main(String[])	1,214,468	100 %	31
ExecuteAlgo.java:126 **lsqtLCS.findLCS**(StringBuffer, StringBuffer)	1,199,812	99 %	944,359
lsqtLCS.java:10	474,390	39 %	474,390
lsqtLCS.java:11	469,968	39 %	469,968
lsqtLCS.java:11 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	134,562	11 %	46
lsqtLCS.java:55 java.lang.**StringBuffer.append**(char)	134,515	11 %	134,515
lsqtLCS.java:47	31	0 %	31
lsqtLCS.java:43	15	0 %	15
lsqtLCS.java:10 **lsqtLCS.findLCSHelper**(StringBuffer, StringBuffer)	120,890	10 %	62
lsqtLCS.java:55 java.lang.**StringBuffer.append**(char)	120,828	10 %	120,828
lsqtLCS.java:47	62	0 %	62
ExecuteAlgo.java:130 java.io.**PrintStream.println**(String)	14,453	1 %	14,453
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	93	0 %	93
ExecuteAlgo.java:126 java.lang.**StringBuffer.<init>**(CharSequence)	46	0 %	46
ExecuteAlgo.java:98	31	0 %	31
ExecuteAlgo.java:57 java.util.**Scanner.<clinit>**()	15	0 %	15
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	15	0 %	15

Naïve:

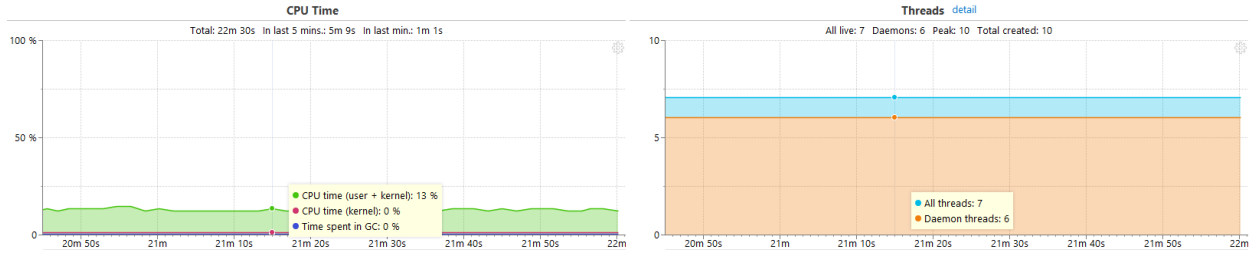
<All threads>	8,015	100 %	
ExecuteAlgo.main(String[])	8,015	100 %	1,328
ExecuteAlgo.java:116 java.lang.**StringBuffer.<init>**(CharSequence)	3,484	43 %	3,484
ExecuteAlgo.java:116 **NaiveLCS.findLCS**(StringBuffer, StringBuffer)	2,484	31 %	2,421
NaiveLCS.java:39	640	8 %	640
NaiveLCS.java:36	640	8 %	640
NaiveLCS.java:17	578	7 %	578
NaiveLCS.java:20	515	6 %	515
NaiveLCS.java:24 java.lang.**StringBuffer.append**(char)	62	1 %	62
NaiveLCS.java:56	31	0 %	31
NaiveLCS.java:41	15	0 %	15
ExecuteAlgo.java:103	1,125	14 %	1,125
ExecuteAlgo.java:126 java.io.**PrintStream.println**(String)	609	8 %	609
ExecuteAlgo.java:105	109	1 %	109
ExecuteAlgo.java:54 java.util.**Scanner.<init>**(InputStream)	46	1 %	46
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(long)	31	0 %	31
ExecuteAlgo.java:93	31	0 %	31
ExecuteAlgo.java:94	31	0 %	31
ExecuteAlgo.java:101	31	0 %	31
ExecuteAlgo.java:126 java.lang.**StringBuilder.append**(int)	15	0 %	15
ExecuteAlgo.java:124 java.lang.**System.currentTimeMillis**()	15	0 %	15

Recursion with Memoization:

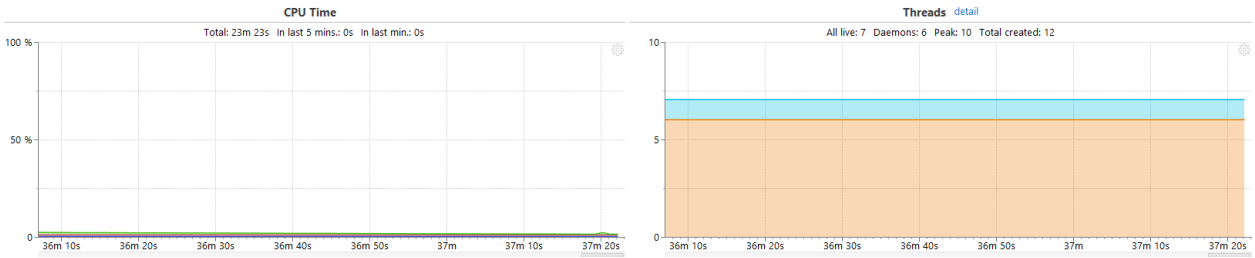
<All threads>	5,421	100 %	
ExecuteAlgo.main(String[])	5,390	99 %	0
ExecuteAlgo.java:123 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	5,296	98 %	0
RecursionMemo.java:31 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	2,765	51 %	0
RecursionMemo.java:36 **RecursionMemo.findLCS**(StringBuffer, StringBuffer)	2,531	47 %	0
ExecuteAlgo.java:57 java.util.**Scanner.<init>**(InputStream)	62	1 %	62
ExecuteAlgo.java:57 java.util.**Scanner.<clinit>**()	15	0 %	15
ExecuteAlgo.java:65 java.util.**Scanner.nextInt**()	15	0 %	15
sun.launcher.**LauncherHelper.checkAndLoadMain**(boolean, int, String)	31	1 %	31

CPU USAGE TELEMETRY

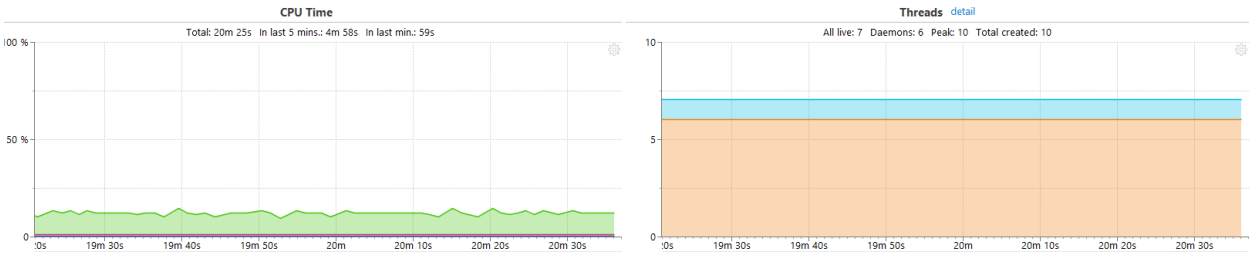
Dynamic:



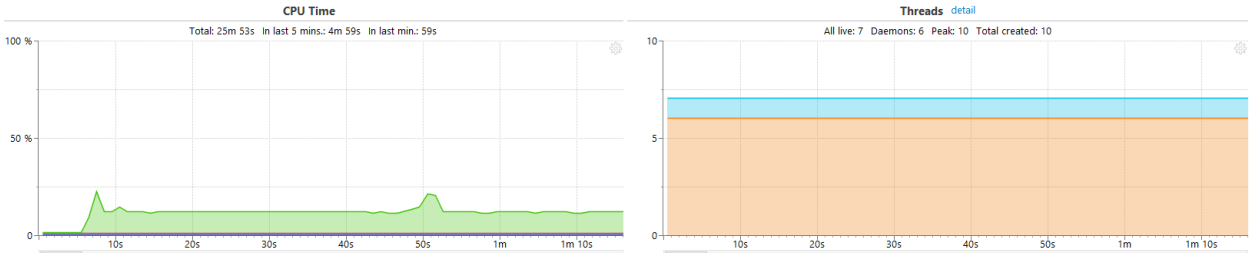
Linear Space Quadratic Time:



Naïve:

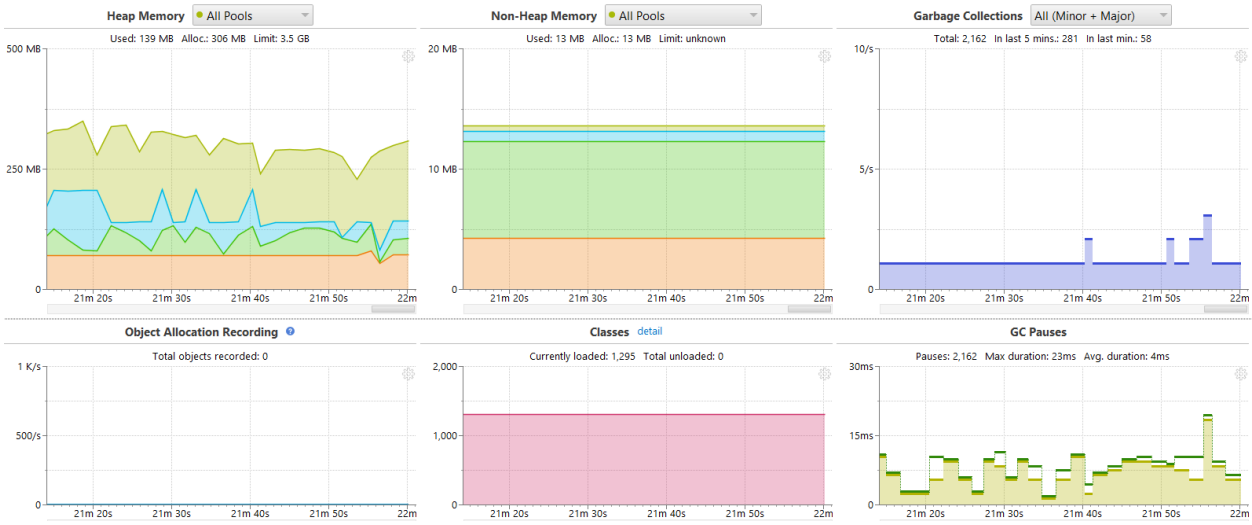


Recursion with Memoization:

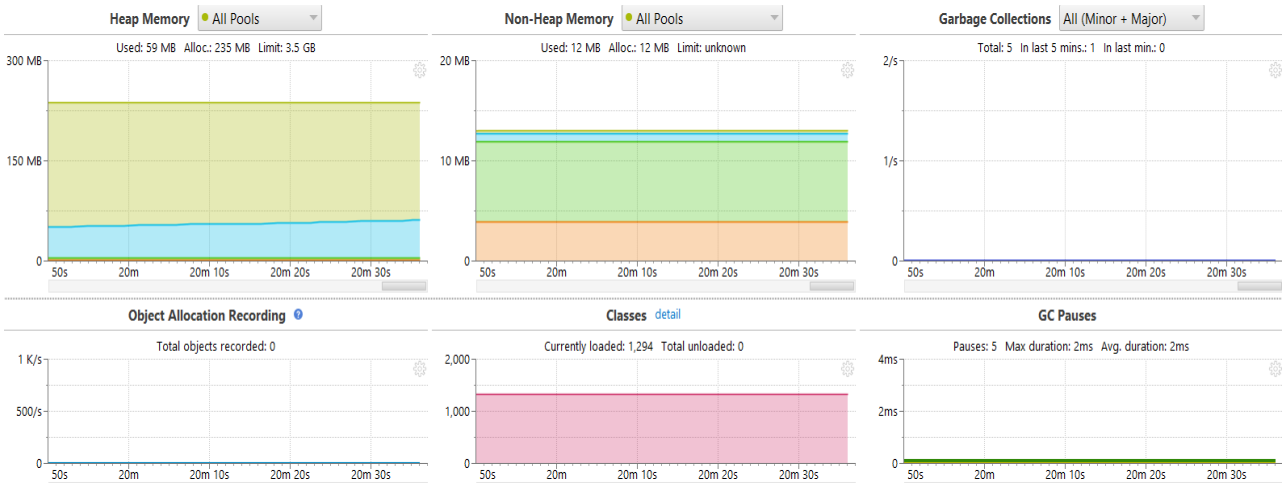


MEMORY AND GARBAGE COLLECTION

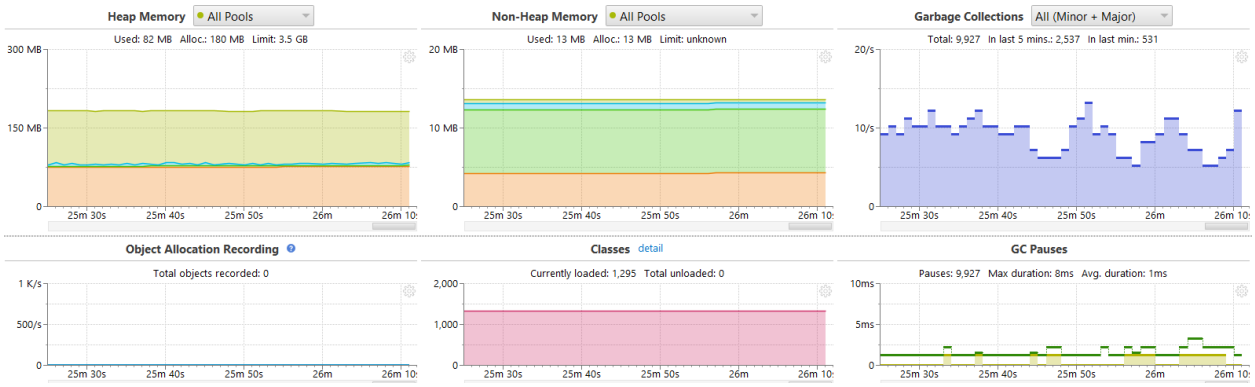
Dynamic:



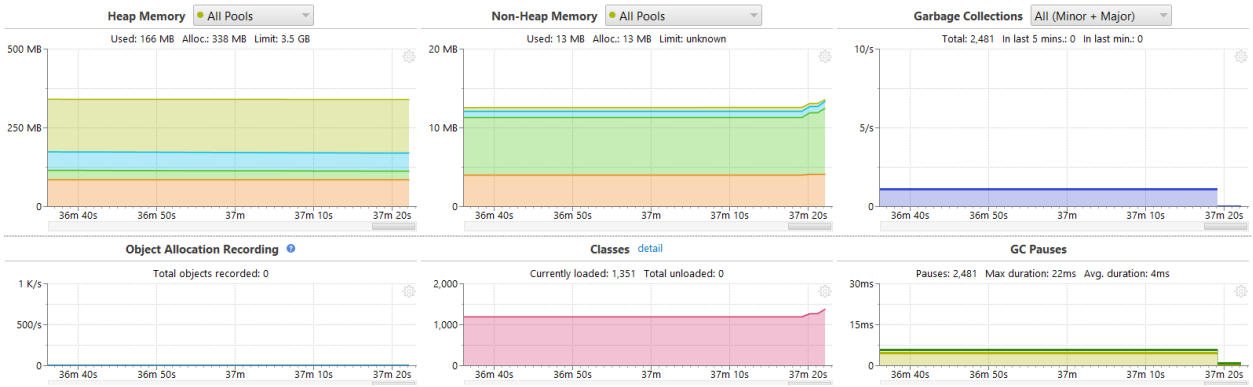
Linear Space Quadratic Time:



Naïve:



Recursion with Memoization:



CODE

1> ExecuteAlgo:

```
/* Objective:
 * 1>.To observe empirically complexities of different implementations of
algorithms
 * for the same problem: finding longest common subsequence in two sequences.
 * 2>.To find out how accurate are the theoretical estimates of complexity
 * when compared to practical execution times.
 *
 * Version: 2.0
 * Author: Aishwary Pramanik (ap9599@g.rit.edu)
 */


import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;
import java.util.Scanner;


// A Class to Execute the respective algorithms
public class ExecuteAlgo {

	// Main method to start the execution
	public static void main(String[] args) throws InterruptedException,
IOException {
		// Finding LCS using Dynamic programming version of the algorithm
		DynamicLCS dynamicLCS=new DynamicLCS();

		// Finding LCS using Dynamic programming version of the algorithm
		NaiveLCS naiveLCS=new NaiveLCS();

		// Finding LCS using Dynamic programming version of the algorithm
		RecursionMemo recLCS=new RecursionMemo();

		// Finding LCS using Linear Space Quadratic Time version of the
algorithm
		lsqtLCS lsqtlcs=new lsqtLCS();

		// To randomly generate binary numbers or alphabets
		Random choice=new Random();
		// To store input1
		StringBuffer S1 = new StringBuffer();
		// To store input2
		StringBuffer S2 = new StringBuffer();
		// To store LCS
		StringBuffer sub = new StringBuffer();
		// Counter to keep track of length
		int ctr=1;
		// Variables to keep track of cputime
```

```
            long cpuTime=0;
            long startTime;
            long endTime;
            FileWriter fileWriter;
            FileWriter textFileWriter;
            int ch;
            Scanner scanner=new Scanner(System.in);

            System.out.println("--------------Menu:--------------");
            System.out.println("1>. Dynamic Implementation");
            System.out.println("2>. Naive Implementation");
            System.out.println("3>. Recursion with Memoization
Implementation");
            System.out.println("4>. Linear Space Quadratic Time
Implementation");
            System.out.print("Enter Your Choice: ");
            ch=scanner.nextInt();
            switch(ch)
            {
            case 1:
                  fileWriter=new FileWriter(new File("DynamicLCSLog.csv"));
                  textFileWriter=new FileWriter(new
File("DynamicLCSOutput.doc"));
                  break;
            case 2:
                  fileWriter=new FileWriter(new File("NaiveLCSLog.csv"));
                  textFileWriter=new FileWriter(new
File("NaiveLCSOutput.doc"));
                  break;
            case 3:
                  fileWriter=new FileWriter(new File("RecMemoLCSLog.csv"));
                  textFileWriter=new FileWriter(new
File("RecMemoLCSOutput.doc"));
                  break;
            default:
                  fileWriter=new FileWriter(new File("lSqTLCSLog.csv"));
                  textFileWriter=new FileWriter(new
File("lSqTLCSOutput.doc"));
            }
            fileWriter.write("CPU Time(ms),Length of String1,Length of
String2,LCS Length\n");
            //System.out.println("|CPU Time(ms)|\t|Length of
String1|\t|Length of String2|\t|LCS Length|");

            //Loop which ends when execution time exceeds 10 seconds
            while(cpuTime<=1000)
            {
                  S1.delete(0, S1.length());
                  S2.delete(0, S2.length());
                  // To generate Random Binary Strings
                  if(choice.nextInt(2)==0)
                  {
                        for(int i=0;i<ctr;i++)
                        {
                              S1.append(choice.nextInt(2));
                              S2.append(choice.nextInt(2));
                        }
```

```
                }
                // To generate Random Alphabetical Strings
                else
                {
                      for(int i = 0; i < ctr; i++)
                      {
                            S1 = S1.append((char)('A' +
choice.nextInt(26)));
                      }
                      for(int i = 0; i < ctr; i++)
                      {
                            S2 = S2.append((char)('A' +
choice.nextInt(26)));
                      }
                }
                // To note the Start Time
                startTime=System.currentTimeMillis();

                // To start implementation of Algorithm chosen by the user
                switch(ch)
                {
                case 1:
                      sub=dynamicLCS.findLCS(S1,S2);
                      break;
                case 2:
                      sub=naiveLCS.findLCS(new StringBuffer(S1),new
StringBuffer(S2));
                      break;
                case 3:
                      sub=recLCS.findLCS(new StringBuffer(S1),new
StringBuffer(S2));
                      break;
                default:
                      sub=lsqtlcs.findLCS(new StringBuffer(S1),new
StringBuffer(S2));
                }

                // To note the End Time
                endTime=System.currentTimeMillis();

                // The time taken for the execution
                cpuTime=endTime-startTime;

                //
      System.out.println("|"+cpuTime+"|\t\t|"+ctr+"|\t\t|"+ctr+"|\t\t|"+sub.l
ength()+"|");

                // Generate the execution log in respective file

      fileWriter.write(cpuTime+","+ctr+","+ctr+","+sub.length()+"\n");
                ctr++;
          }


          // Generate the output in respective file
          textFileWriter.write("----------------TERMINATION-------------");
          textFileWriter.write("\nS1 Length:"+S1.length());
```

```
            textFileWriter.write("S1:\n"+S1);
            textFileWriter.write("\nS2 Length:"+S2.length());
            textFileWriter.write("S2:\n"+S2);
            textFileWriter.write("\nLCS Length:"+sub.length());
            textFileWriter.write("LCS:\n"+sub);
            textFileWriter.write("----------------------------------------");

            // Closing the Features being used
            textFileWriter.close();
            fileWriter.close();
            scanner.close();
      }
}
```

2> DynamicLCS:

```
/* Objective:
 * 1>.To observe empirically complexities of different implementations of
algorithms
 * for the same problem: finding longest common subsequence in two sequences.
 * 2>.To find out how accurate are the theoretical estimates of complexity
 * when compared to practical execution times.
 *
 * Version: 2.0
 * Author: Aishwary Pramanik (ap9599@g.rit.edu)
 */
public class DynamicLCS {

      public StringBuffer findLCS(StringBuffer A,StringBuffer B)
      {
            int lenA,lenB;
            lenA=A.length();
            lenB=B.length();

            // To store the LCS
            StringBuffer substring = new StringBuffer();

            // Initializing the LCS StringBuffer
            substring.delete(0, substring.length());
            int i,j;

            int[][] Matrix=new int[lenA+1][lenB+1];

            // Initializing the Matrices
            for(i=0;i<=lenA;i++)
                  Matrix[i][0]=0;
            for(j=0;j<=lenB;j++)
                  Matrix[0][j]=0;

            // Loops to iterate through the Matrices
            for(i=1;i<=lenA;i++)
            {
                  for(j=1;j<=lenB;j++)
                  {
                        if(A.charAt(i-1)==B.charAt(j-1))
```

```
                        Matrix[i][j]=Matrix[i-1][j-1]+1;
                    else
                        Matrix[i][j]=Math.max(Matrix[i-1][j],
Matrix[i][j-1]);
                }
            }
            i=lenA;
            j=lenB;

            // Looping through and finding the LCS
            while(i>=1 && j>=1)
            {
                if(A.charAt(i-1)==B.charAt(j-1))
                {
                    substring=substring.reverse();
                    substring.append(A.charAt(i-1));
                    substring=substring.reverse();
                    i=i-1;
                    j=j-1;
                }
                else
                {
                    if(Matrix[i-1][j]>=Matrix[i][j-1])
                        i=i-1;
                    else
                        j=j-1;
                }
            }
            return substring;
        }
}
```

OUTPUT

CPU Time(ms)	Length of String1	Length of String2	LCS Length
0	1	1	0
0	2	2	1
0	3	3	1
0	4	4	2
0	5	5	0
0	6	6	1
1	7	7	1
0	8	8	6
0	9	9	2
0	10	10	8
0	11	11	8
0	12	12	10
0	13	13	3
0	14	14	3
0	15	15	11
0	16	16	12

0	17	17	12
0	18	18	13
0	19	19	15
1	20	20	15
0	21	21	7
0	22	22	17
0	23	23	7
0	24	24	18
0	25	25	8
1	26	26	21
0	27	27	6
0	28	28	8
0	29	29	20
0	30	30	8
0	31	31	7
0	32	32	10
0	33	33	10
0	34	34	13
0	35	35	9
0	36	36	7
1	37	37	10
0	38	38	14
0	39	39	10
0	40	40	11
0	41	41	31
1	42	42	14
0	43	43	13
0	44	44	35
1	45	45	34
0	46	46	35
0	47	47	13
0	48	48	36
1	49	49	13
0	50	50	36
	. . . .		
8	458	458	372
9	459	459	143
8	460	460	144
9	461	461	145
8	462	462	141
9	463	463	371
8	464	464	148
9	465	465	371

8	466	466	368
9	467	467	378
8	468	468	373
9	469	469	375
9	470	470	368
8	471	471	150
9	472	472	380
9	473	473	147
9	474	474	146
9	475	475	151
8	476	476	374
9	477	477	379
9	478	478	152
10	479	479	151
9	480	480	156
9	481	481	384
9	482	482	146
9	483	483	379
9	484	484	382
9	485	485	151
9	486	486	150
9	487	487	154
10	488	488	393
9	489	489	387
10	490	490	151
9	491	491	155
10	492	492	391
9	493	493	398
10	494	494	152
9	495	495	150
10	496	496	156
9	497	497	397
10	498	498	398
9	499	499	157
10 12461	500 . . . 89618	500 89618	400 32507

String1:

HYMZXKIVDBYAWMQDTTWUJECDLWLVAINHYQYKTRKQJIICDVOGKWAURTHJFBKMOSEZTQQEGBQGWGSDNPWHZLEYHKUZUQYCTBEEMXIWQILPII
LFZWLBIYVNETCDEWAAFRTEKAEZSKOMLJUZGIWQZJIQNNXGFAPPMMVRZUYBWUTVZNTNUNNDECWWNMZLNLQRLQFTLRMZAZDPUZKTJFMYZPXM
HHZFSCTVBMWHIGQRPNCIGWKPSBMHYTTYURRISDSCTKCXOWIEPLYNFXQIYXQEJUHYBEAQJVYFTGATNZWFPJIPZNOXAHGRCDCXTOEDZVDPWS
QODWDJJSAZMUSFZTYFFRSYZCXYLUREXCGQENTPYEJFDHBAJHPYKXHQMECMHXWCDXDGWIFNIKNHGHIDWINXHIILXKQUQSXYUCNSHDERCTQU
KDIXJCAMSEGWTSLYIQKBUDFWLKQAJVAMVKLKPTPRXKVSCIVQEKVQQJJMTSDFIEVNEHPHLFSDOBWLDEVCKTKBTVNAORGRWBOQVCRBNMBCOG

IYGYFSPAIBGZGZYYELAOWHVWUGUKAKBQZSALHBIGKDYKZCIKTVVXGUBGEDKMTTAHGKPBKMHOLBJFKRWUMGMGXVRRPRJAYOFMNCGEDOUGZU
VMXBULILAVBKPQJSSIGVZKQJARINNFUTJPTLUHRGLGYOHFGPXEBUKSDQQNPZGQWDQECEJLHNIYDWPHWVQSRQENUXWKDCPNTSDRFQFKNJFR
MQRNEFEPNPZTHPWNQDMLVCUDGFJZWQRHLSNKFHMFDJLRBYHIGBOVNKOSLFLLGIMUECUPYVNINAKOWBUEJVCMDYRYNVBTAZZYGIJWJXKKAF
GPRBFTZDSDSUKVFHXBRFIWKTAQKCLZJJHNVGQQQTJVBKHDLFCWBEGIAWQQJWAYRYZIGWYGQHWFWKTFRWEPSINULCVDPPWBJOXPZMYJNGJR
TKPPKOKKDZKQFDGQZKZLNDSFFADFDOJILSKBSOQLPPIQMVZJCQHNNNCFEGLHIDOHBYBLLYEXSWIQVOHDOOTODBGSSPPIEYWOEGFTEHRUTQ
BDTXSJDFDZHPVNUOVECPBGIAVUIUFVGQQJYBGYXPAUUFLBHFYMOMKDFSMQZMBNZMQIWZPPCIZGASXYJHSCWZTIGADXJFKIAULUIPRXAFVB
BCTBXXMTDBLDCSETRXSNLDKRARYJSNWKMNDBNJZTNVKYROKAJLMYOYZYSBGXOFGRUULMJEANAKFMNVXTLJLDHVFNVHDGVSIBNDZJBHUKQW
YNQVCIKDDQAFFSYKXFUQNXNSTANVQQFFQYQOZSPQIVNPUBFLUWRCGZTSRVJBVFVLBLJJHWAPCJXGEAMAEILLQBRLVMUVEWXZZSMPMZPXVY
YXSMILKASPFZXXNDHHOZJKGWOOVYLRZFFKLRROYBSNNNGUBXRSDABRXFPXCTWQONDFIORLFUVGTKRPBYSJXHOUSMHDSTTBRXIBQKXHLCDY
WQCLVDOLLOAJOTYHUIYRNCSYCMUWFSACPNXXGDYXJBDWVPCETKXCUCQCFYWGNIAVUFXJCXYBTIADMVDEOKGARSFXSTFWEJPDGZYTSYBCDW
EXZHGYUMVPONNAGNMANCHTALLFQQAQWPDTMMFAETXZRAPUWHRWSKPONCYAZURCCRVWBFFVYMBVRWYYXSXTVYXEHNVVJPNDNNNCWHTWMNKD
MZSQCOKVWMKMRDFTTOORGLYNTBXXPEYXTCTBOISYZHSTHVXCEGRGIMMDMWARBCEEMCEETRAPAIAPUZPUQFEYYYZTZXZTADYORIMDERVGRQ
DPZWXRBWMDPRGSCGOWLXCEYLPWLOJXJYJAHGQNNKWUMDLHOOBCQCLWTDOOFVMRJADBFBMRGQKWUXYQQKZAQAACKVSLSVNWLBDZHWIBEVNE
EFEFUSNFKTTEZIRBHAUAOOAUWHWTJQVADYIOPTDPVCISKJDVVJIIYUUECFQCGXKNPCIMVHTLRJUAKRXWJMVKGGMQOFNEGBNCBTUBPIYMEM
HADHEWQONSQUVFFBDGLBVYRSATBYDMLLXXOUAKDZZEXDJAMDJOSITDQVFFUNKMFKNEZCYWRSSZBMHIFSMUDXRPORYLQYKRQSKDDWOZXCKH
SWTXXYKPHSJ……LPOBHHGFKOBDHTBKRBPVJYNIPQYVPLWAXHPUWYMXTSTCNREBXONHYQUOGXQNSDIIRCJBUMHFRDTKIXKBSUPOXVVEIDK
MPOGRHLZHXNCBIUJCGICCLYXJVTDTULSSAERJKLXLWFMBNQJCIMOBANKGAGFAIJKYXMXJPQJWPDCCLMAWPMWSKASMMEKPLMDHUUFKERJNG
TCBSLWXVNAFOIBUEDVJUNJPVVBOSPZRFQVIJKXNMBOSAMMCQVFMTDMHNLXWIBRXDOYHHNFRGFJOBVAWCAPZVZWBZDRUPUVMXXNTNJCONMX
EUVDQSKTDLUXRFZNGFJYJIRNBLMGJWUHJOZITYJCGQSOIRNSGXHQIQNCHDTKFPZMYONTUVDJGKLVMBNEYYRNMRBBEVSOZYDJDEOCTOYIWA
RUCDZEOUCRXHCCZMAGZMATNEMTGRGOJVKXJOJNNOROFTKYOUTEDFYAYFSPJADSBHTOSJDQCBJRHARSJFTWRXNFTOSUDDWXHHDMOOTQQTHS
AXFGTYHJBXTRDYCUPUBWHYJLGEEHHFIXGMDPVYBRLLYOKGPJTYVMKAHRLQCKRUMQRHDCZGRWZFHPAYKXRJJAXOMQIJYONICBOGKKAVGIGM
AMGCUTWLPQEFISUZWLRAALHLXMIZPVOAECDZKZZBORDVCHCQUUSNNMDLCUSIHCPLHFZQVRGVJQRRYIQWTTJOJGQDBTFDZXOCAPNSYLFLYM
HOZHWQVPOLHPOEKIBRPMZTLXMAZPXKXWMSZOKQDYGXOXYWVMVJKPEYAHIIGFYUUWKKCDVKGJFFVZQHIYLGGORKDYHRANILHLTFVYIDIRIB
IDNTJBLIEQWTDAMOSEUAACWRFUQNJSPHXDQVYGSWXNTHEBYFLENTLGQPLHMEVFRMHTBGNHLDLCJSZGFDRJHIBKPKUJTJYTPYHLKPVLNJJY
SFRBAHJKGKZOZLDUHYAJXTBDZKDJQZTEEEMDAZGRZZRRCKTYFVLCWPLHXBJTGLPTPXLWPAPFUFUIALOIXPXQWUKIYMOYEGSTMDGNHDFUVJ
PUZCOQPUGERADIUCPYZNLNUSXXBOFKZHSKWYQVWGFTURZLPJEVLQZVPCMMMSDXZBUEVBCUNPOCGXGMTXOQJPRNUUJRWSNEXLHJJNHYOAYC
IVJVIZZKAMGIPORDUCQNPOLRFFROTZFWWVESHZZPVKJOQYWNGBLZUIOEFPPQDSMVOFGTSTASLYXISJDMXSAYBXAYWVJVAUHVUAZAPNIIUM
HAZHVZCOLANEFLRPELCDAPILEVVCPKSPUEUGOPDJBSNXLQEOLFMRYSYWICDVLPJFBGUYBMEPLQYWINJNXXHNFQLMWQTXQOKNAJDKBFUMKO
MCSXBTBCHJQINMBRPIOTNYYXLEQZORCWXLJRLCOZYZRXBTCLXTTZLEBTOYYIUOOUKNZBTTNUQUYESMSYGRJDSCGVRVZPNNBQATGGFWJXKU
RYATACDAOMFVNJMBXILBSSUHYYGAHOJLQSZAVGGUDHXIBEZTTVLZBXPBELZIARNEWLUSPORCCPOULIOHSVUULTGYAFXSPFPPJFOGOQPSYY
TLSNNXIXYJSEFWJIKCOIXYEDNJNGUMRNGHVJQVYPHQZHUBCDUOIKLCHUAMMSYOWBLXHGMFBBKKNDKKNI

String2:

IGXBALOBGLHZVOTIVEXWHOLZMTPMAQZGDUSYQRYVDSDHXBMILZOFBMNARQUREYXIJYRXOQLSIUKAUTFQKNTSPSNZSCYCCDZDHLXNMMMZDT
XUSOEIMOZBSYQIEZZFSMPCSMSBGIGXIHVFZXXPXUZYNEMJBICBLYRXJZKENBNVFOJINZUSXUZMDJEIBUDYMYRTZJMNUTWQGPEPKMLPKUKW
HFYGHSWFTGVLBWRGBEEWTXLWWDGVEAOMTCCIKSMXKGBKERISCUBAPPZDMRDZCKOHCYQQOHPXGACNLTMVQDPTXTFQQNEQSBIHGIISMGMOQK
LCYSGPRMOGTMADCJAPTCZFXKURDNKGAHCIFXPFPQWEBBZPRTUTGABRUUWTFLIKGDMWIDRTFMKJGYXUCFWPNVCSZXDNWVKIKSMHQHCEAYPU
TIESNCVJQMRPMNBERRUEHFGGKQNHAKTDLKRNOOJCWSSWSIJAXWPYUGBVTWKKRQRKITYVEJCNUXFJSHDOONWRVMVLFZUZSLKHQBADPYNPTY
XXAYPLDOIJIDCWQGRTLOCGGONHRXABNADFUJSOWQSVEBGWMEEWYGGUFMMLPCULSVXHVHSGAURQLLNDTGXGVFIZDPVHKDRKDACYWWKMQZUP
ZNXRLNTDWLCOXACXILLFFSACJUZMUZVRUBEDDBNHZASKLYUFVRQPMFMUATLVQXSASXOLJVGYMNYJVFVUGQOXEEJGOQWERCIXLDRWMOJKJU
CKPMNMNUNRXSQZPYUAOGPGPEJTLSNSMZCDHFNNDYWEDMVWBCHARAMFKLUCYVBCPPYKNLGDIXTRGVFJFQJCRNDDSPWYOMYQCSWUSFWYWXTJ
XFFSMEEJEBXCBDZHRRBHOUMQDSZHCIWVDCSECWSUZXYFDIWHMQPRDBOQTFGKPFITAPGBFVTMJMHVHXBOUHPDVIECQABLBOKIYHGFROOYGQ
XYTNBOKMHDVEGWEPKSKBUBDDQCTSSZCUZITCKMPRSDWIIMCTCDBKRVTPVGSKXBURAWOSREZGQFGOHSIWNJIPKTKUWLCOKJBULQZOKXOZCU
JRJNLIESTRVZTJHNFKKCXPHKISJOKUIQFKZVQRPCPOVIWYYTWQZHHEVKHQSIITRUZNRNNOAZDWJWOGILYICRWOAWOBPASQMYYCKEDSUVTC
VNKABPTIJQCDEASFGWHGOMYAIBUCBBBRLXMNZKZQMFHBIHAUVBMTXXVOFPVINJAJOODHNFKJTXAQWFABEMNNLFEHRBQNDFQNBTEKZKXRGR
OVPAQAFGQTRSXYIHUGADTMYQLPUIJBJARLRCJZGOSGCQUAGWMPTDADDAESQRKZILMVBNMLTUWOQLXKLVUPFHGQSQEVIFMDDEIRRYRILBEA
BNKQFIXFIKBJUEMMEFCTIEJCEVKIIWNZESNKSFZVUUJWFDWBVBITBGJNAWXCNURYUHGJTZWQYRJBWBEMOGEYIKBEYZXQAWPOMUADBMBHKJ
FHNCHXMLUBKAEKZIZWOUGJSPCFMHTYKUUDDFQWORCCBBUEETNODBOJFYJJLZNYDTFPKNRNDDMRMDEXETHXTHODXFOHVUDGHWYVEWBYGHQG
XDITQMURUQKEJTYUCRDIPHVNLYJVQISBRUJWNCKIAEALTEZCDDRJGMSBRZLIGSUPYYSIZNJPHHBNEAYVCDRSRVSEKFFHLVCIHIPGVRYUJC
USDDRVXATGBEVEEQCYCBWNZMMMTNMHIHCDFRWUWTCNXGZBRTVWBRLGIESAYBRZIBPGJZQZRFEQZXUIBYENLLVPIUGHQFFEDPHAUFBNSSWU
BRYNEAQBMXMJRCSUCDHSLVPGFXKTFPFTLDTDMDIZNFVSNDRSWPCGAYIERXPIGPFUZYVGTLAEMTJOMTRTROOMAMFGBJCXFCGMEYZKWCWQOS
KJCTLCXWPKSTQSOWWHQTHLNICUNARIMVIR……AASYVYSPUZXMLKQXJEROZINHFBKLPELTJLAHOXRBLAHOEVQJGSXCWAPCWGXYESAYBTV
UXSMCVLHQWJORKRHHNDPURARSUCOGPWQIDKMGLCVOFQDQHLVIANJKEBFVKWXLUZSYFJHVFAHAQYROUVMCUTAVUNFGLLXTSXHKAENLXKMKT
ILLFXTQXIGXCVRFQQHKQFOZLLCQHRBWPYBSHYEYEGZMOVQJQNQLOAKBDBYDZAGADQNMQMLVXTKLKIJTQSEGAOBDWARZNXSQACUGVNCHIAH
EQXSDKVRBGINCGBARRDYNCAPENADLQNFCEEUIRINFKHVEQDDLRNWAVLABEJUACGNPRKFLDGRWEQGTCAAPRUSRPXDLIBYEDRDBIZJBFLUXK
IGTISWBDFBZIAFFTYKVMYRNPJIUPNUIQLKKUPHJNWMQJKSPRZYFZRTFNGWNAUVNAITCJVXMEOJVTRBJULSSIVYPDAHBEFOZUETPNXKKOSZ
PHOUKGUJYYMJBRWBEFYQUSAMHUNAJCNRZILTUGBQZAGANIEWPOYAGCNLKRHVWGCTFZHXVXDSDPDZYWNKFUGZTSOHSPDWJHNROSHQDXRZAZ
LAFGECSEMGRUQCHDUTPHJSBUITFVILPTVKQIWSFHGNHTZNDYSMUGOJQHDDCVKBWFPGNNCVZEHSCPMANJLBYYSOUSEQRIOEHINIONEEWUDA
JVUSBXFRMJZIEXGOKEWIPSVITEFXHJGOIWLWBSEROWVTRXLFYSAAJEPPNWYMVZSVVSMYWAPFMSTSZFYUKIFGJZAMITEMCPLMCOGUZURCWE

VUOMFKAAGYOSEHADSEFUGUGBMTKUGESZCRFJTTGQTMZCEUOSMOFBLDEHZGYCJHXBFFWBDPQUXWFCZLWYLNNAFBRWOHINNWGAAZLHNWWEXI
XNCJPEOABYVVJQGLLPONOWKSWGTDNKMDNIIVHMSDBGGQKBDOWVHVXDRCARRVFSTGHEFRWNOGDLTMSOYJNALVZNLQERXCXYNQWAMNPLOUIZ
UTGWAFKHCUABVILNIIBAYBDLLKPVNWESAQUBECMFSRVJMFZDSCXMDHRZILZNMCRJZJCSJCERNTPYKYNCZXEXTPTPAPYWODMJBEJMDGFCDK
ZLGBSSVPCFNVXNTTBJKUGPJSVTZREJPKVWLJYURYHVYWBOOOLCWROERGGYQPHNPRTZKQEYHKRYUFGMCKOCFGHYKGXVMBZVDVDYKQRJUGHI
WUEOUSWIPUXHNGECPGZMLOCYUJAYTDHIJGVZHAIQMGTASFVESADRDIUQGRVONXFNUFZURCUYCMJMFTPVRLVTUYLFVISKBSQPIULYHUPIJM
QEDMSUPGMMCWQMGDXTTNAQJVMGHLBEUHXOEXYNVQUJLWLJIHNIUTHYOZBLDMCFTKKHCHVAODXKBHYOGKMCMKMDETWAJFFSRYTRRONBEWVS
OXMNSOWWPYJNQTGWRTGZRDUPVATCYQSRXYDZDVUPKDQYFGIOEOPVPLFRYGYRDPGYGVFTEPXMDYWMZOWLAISRCJNYVHMFOXNOHRTRUGUWBK
BTVHRJVMDBKMHJKJZEKHVEUHJIWTJOBXVCVQUZDSZMWJYFHYXTZKQZPEPQTVNSJHKWFIPBGJBAMUOXWURQCQVKMIPJNMNHBHPZRCNENRJN
OXGEVYPZJUFUKDOXRURDWDCELCLMUUVTRIJVZXRDAIIINNKWQOOVKVTRKMAMJJZCLEZDSTAEDJBNDEXNSZJTPWNHISUWSDTQKZZHUEYXIC
YWKHSUDILWHNQUAOGDRYUPRBOJFSFNSNSQYOZYMHUDKACBRCBXFQKRTYNYVARCJWOHEOQTBEIXWFKQNNBJYSROMZAOCZVOMULQMJSKRIGT
TJGHTTROPWMOCMLSFZXBMGTBILBNZWMIGOOADZJFXVRLJSOKILNXSWGHWFVZRIARNXRPLSOXLJVPCTSVFVUOWHUAUHBUEGOVQJRGNQVYLW
GEKCPPMBGENVQHLOBOYVJJGWWXTJDYSVAOJMLNTCFZTLMNVYZPFYAENONWWGFKINQCZHFDNGABCCGDQZLAMOSLMDYSKMXGIEHMLUAWQKLT
ISDPEOMTELNKZCHPYIPEDIKLRJVPCWYSCTVHDWPTHVSICPVOFAWNHGIMHGFRJQQMWGGGYZDLCXIJRUTXIQEBLUSTRSKXRZPQMASNTLCBGA
VIRBGOAXHDOGNVKKQYDYTEBZQTMDSECDQFUIRMVULZUVJLUHIJZHZDEQVBZZJJTGGVLPNFONXKDAUPOXBPTKRBRSGVUIGAIFFHIPLWAUYE
VHHZNKGTHSDZUDDNTZQZNIUCLMKYWRZGICSBKUEVPBMVIBNYKNIULUQFJWYBFKWZFACFVXHSWAEJLZXQKFDRJIEFKEVXNRIGUJZHRHQWQP
BJHGNCHKKJGLOLISXKLBACZPHNGVQUXAFXDOOYLHYNVUNIRUMWAVMOAMTAIXPOZWFDNUZDFHERGFOVLXDLFRJPSCHZJIGXQJPJERWHUBFQ
CKPVEIJJZDINXKRCLXTQUQHPMFILWLTVILXNKPHKTILSALZYRCVWMUQTSSQMUUQWHNWMSCHIGZHFWWVDLUDTQZAJLNGVKUXWOLVXOSXUEH
ROYKEWELTVUJLABDZVZNOZGQVWCCJVPSNNRTFJZPXPMQYSNOYDVMAXFUTMYAGHRKE

LCS:

HZIVWMQDUDLAIYRQIKAUTFKSZSDHLZUEMQIPIIFZYNECRKEZSUZJIMRTZNTEMLPUKFYHFTVBWGCIGKSBMRDCKCOPLFQQEHYAJATZFNAHCX
EZPAUTFLRTFJYXCWCXDWIKHHINHKQNHDRJCSWSIJAVKKRKIVEJJSDVLFSLKBTAORGRBOQVBMGGFPLHVGUQZHKDYKZCIVUBEDAKLFVRPFMU
LVQSSGVQJRINNUPUGGPESNZDHNYDWHRUCPNTRFJFQRNPWQCUFJZRHMDHIVSECUYIWMDBTGIAGBFTVHXBIAKHGQTBKHDEGWQZIWGKRWESIN
ULCBOXZJNRTKPKOKKZQQZKZNADOILSKBPIJQCFGHIBBLXQHBIHUBTXFPVNOFJXAFLBDFQBZQGSXYHGADJALUAMTDDESRKMBNTOKLMYRLEA
NKFXFIBJUKWNSKSFZVUBURGZRJBBEMEIBEWMMXMLKASPFHKWOOYLZFKRNRDXXTODFOUVBYXDTRQKCDVLJIRNCCMUSPNYDVEKCUCQCYWNIU
CXBTIEARPGZEXYVPGHFFEAUWRNARCCVFFMVRWYXYVTMOMRTOOGBXEYTCTOHTHCRIMRAAPUZQFETXAOEVGPWXBWDPRSCGPWQKMLOQLVJBFK
WUYAQAVLSHENKTTIRHOWHJQADYDDQNMVTLKJQEGBNCUIAHEQSVBGBRYAEADQFUNFKNECRFDRPRLYRDZXKSWTYKMNJIPIQPJWQZYZFIJJVI
VZEPZHMJEYUSNJRZTUAWOYGCLRGHVYNUTOPDQLFGCECUHLVISHHSGQKVCALSRIJBFOIJOWSERVTEWVYWFIJIECPRCUOMKSEDSGBKUGCQZO
DZGCJHXFBWFZWLAFHWGIJPEQGOOKTDMDNIIVSBDOHXDFSGHWMNAVNLERXCNMNPUIZCUVBBDKPVWECRVJFZDSXMDHRZINRZSNPTTPPWOKGC
VXTTBGPSVRJPWUHBLROPHNRZQHRYMCKGKYQRGIWEWGEPGODHVIQVAURONXNRCJMFRTIKBSUPEDMPGXNJGLYVUJLLJINIYMCCAKMMKMDETW
AFOBEVNOPZRVACQRXDYFGOVPVZWRVMXNTUKTRJBMJUHJITJCQSXQQNHKFPMOUVJMBNERNEVZJDORUCECMTRJVXNNOOTKEDSADBDSJTWNSU
WDTQHYXYWHLHGDPBYOYMKARQKRRCWHOQIJYOCOKIGGTWLFIZWMIOADZRLSILHFZRRROJTFOAHOQVLEKPMLOYGXYVMVPYAGFKCDGZLODYHA
ILHYIDIRJWTDWPHVSWNHGHFRMGDLCJRIBKPTLVRBAHKKDYTBZQTMDCFVLLHBJTGLPPXPAFFILWUYEGTHDUZQIUCLKYWRZCSBUEVBNNUUJW
YACVJZKDRFRZWHKJGLOLISXAXAYHVUNIUMAVOAAIPUEGODJSXQERWCVEIJNXXHFLWTXNKKTILLZYRCTUUNMSCGVZNGKUOVXSUHYJLADZVZ
ZWCCPSTFXPQYSNOYDUMYAHK

3> NaiveLCS:

```
/* Objective:
 * 1>.To observe empirically complexities of different implementations of
algorithms
 * for the same problem: finding longest common subsequence in two sequences.
 * 2>.To find out how accurate are the theoretical estimates of complexity
 * when compared to practical execution times.
 *
 * Version: 2.0
 * Author: Aishwary Pramanik (ap9599@g.rit.edu)
 */

public class NaiveLCS {
	public StringBuffer findLCS(StringBuffer A,StringBuffer B)
	{
		// Variable to iterate through the StringBuffers
		int i,j,k;
```

```
		// To store the LCS keeping A static and B dynamic
		StringBuffer sub1=new StringBuffer();
		// To store the LCS keeping B static and A dynamic
		StringBuffer sub2=new StringBuffer();

		// Initialization
		i=0;
		j=0;
		// Loop to compare A with B and store LCS in sub1
		while(i<A.length())
		{
			k=j;
			while(k<B.length())
			{
				if(A.charAt(i)==B.charAt(k))
				{
					sub1.append(A.charAt(i));
					j=k+1;
					break;
				}
				k++;
			}
			i++;
		}
		// Initialization
		i=0;
		j=0;
		// Loop to compare B with A and store LCS in sub2
		while(i<B.length())
		{
			k=j;
			while(k<A.length())
			{
				if(B.charAt(i)==A.charAt(k))
				{
					sub2.append(B.charAt(i));
					j=k+1;
					break;
				}
				k++;
			}
			i++;
		}

		// Return the LCS
		if(sub1.length()>sub2.length())
			return sub1;
		if(sub1.length()==sub2.length())
			return sub1.append(" "+sub2.toString());
		else
			return sub2;
	}
}
```

OUTPUT

CPU Time(ms)	Length of String1	Length of String2	LCS Length
1	1	1	1
0	2	2	3
0	3	3	5
0	4	4	3
0	5	5	3
0	6	6	9
0	7	7	2
0	8	8	7
1	9	9	1
0	10	10	2
0	11	11	7
0	12	12	10
0	13	13	3
0	14	14	3
0	15	15	7
0	16	16	9
0	17	17	7
0	18	18	9
0	19	19	5
0	20	20	4
0	21	21	25
0	22	22	5
.	.	.	.
1040	172503	172503	6710
.	.	.	.
.	.	.	.
.	.	.	.
7234	45229	45229	6213
5023	45230	45230	6710
13249	45231	45231	1800

String1:

NSCDKQYKMIESGEEOHWDFMNWHPHOYWTJHMRMHWPPLWUZOKBIBXCVPQKYHLXANLYUXKRRNWVXFANCDDNCSCWIFIQUZY
FUPSGRKFMPFGLBJIZIDKYCOATOERAXMFZKSMMHBLHYZOSNKQMYSKCQKGIQKJHWQZTEAELTMSJPXUKSPTTWYKAVKGE
MVFPNCYWENLXOTPAMJZXHFJWTLYIRPDDYDEGDIBFJDWQODGALVNKTRUWKENJAJGLRCJNDZAKOMIJAJWJPBPURKBNI
NMDBRXDHWHTEHSNBAMHPXAKCJAPVPNFMJCFDXSMMLUTHVJAQXQHNZNYMRDFLXOZVOYAIEAUSBRRMOSOVSJVXGDZMJ
VDCDTMOQAKLNRPZKBHBPWEUZAJKYMPKJYITCEEDMLYRTNIAXQINCQQMJTKIPWDYVNEPYNHCWVINVCFIBJLFDZWLTH
FWQVIMPJUVCJGDHGEOAWKFWIIQDFFFXMGLZXKOGABFEWXAAYFOVRWLPWHHZPWBKSDRRWXGYNDTNRBGBHGDZPKESDG
QJXFLHKBVCUPYKMYCHXUIYAQCDBUACRRQLBSGUHPIPYTPOONGLOSBRJZBORSCHJKKUNYUBHZMWQOKDPXLEEKDVNXA
WVSWEJIVJTPONDVTMNDDBIOFUSELMHNNRJFWYGAPAFAYDOSYOCYSVGBGPUEWPLGJROKWBNWFSSXVQRGRDBCOETVWL
GHKODYIEXAGEQOPBZBMNIYVSVVXVAEECNBEWLWNPLQEIIQIZOPLALQDORQFLFVSCQAZUHFCWZZVKRRMAKTPGPAUNJ
KIXWAWFFWHJARPOWLGRRRBAECFLSVMGEJMHVZGHDCRNOKDLBKZUQMGZISVLCQPDOPYESNXQLNOARIJQFPVUMWHPFW
YTNHVCLJJESFYSJBSJEGDYGJDPYRDCADDJOEINZMRWPDQXUGWFZXWBFSSCQLQSSNNHDENQJXLRMNTFIGRNLHXVZMS
LKGAZODUYNBLBAITOUFXSUDPBTRNGWEIHIWJPVBPHHWAMJHEQGQQOSHZCVBCITJNYWRFSCPFGBCCPTBPELTTPJZUR
XFQDGQXZBVXORMMXVDRUNBDPBUJIIKTLCGDZFRLENELCBXTJMHDTWHLDBBDDJOZIWMCDRIGLQFEOROTUKKPYWLNPP
NLXBPWWSODMPGGXQGKZXYEQNVRXGBJHFIJADJIUHEOKJCERJDHJCGJWDKNTTDQSQMVZWEQJSMUMHSTSSJYFKYJCKD

KGQHYWAXNORQHGKFLBZODZMFVVULAFGAGVWWJQOVCFGWMEJMXZKIDPDHEFKLZTGFEUHNRCHYZOAZWZQPKHRMCFERX
PCCZOOO……..SMVFBKZFNYNNWJVVSQIYRKZCSNUEURDLGXMNMQBSBKQWQOEZYTUKOCPORAIJY
YDLCLHERADHJORNRUSYGYURPOREJWEMNLTKFXQOPHAKQAFAXRMLVLCIASNOGUSISLCHWKJPTBFCTIORRGDGYCQWMT
YFSOIXIPUJTSIWRARKARYEZAAXOKSQROOXGKCUJDYUEUYWJJHGTWVHASYOGIAILMSFPKMEHAQQASULKPQLZLNTZCW
TWTPDKXQAXJMHCVDPCXAHDADOCSMZQJZGWDCQAKXLKCKABWVRCDNIYJTVUJURCESGJVLZPICJPIWSXQJVKMQBSFHZ
EEZVXRKJIWTFKLKSQJOGXEAFSRDZGPESZONKDBPLZQYPUSNTPPYZLYDBLMJQIIZXPHFAXZYGYJBAEIBMNQLFTDUJK
LQKAEQTQDRYAORPUIOUROOZWMRJPNHODEMAGDBMMADSUYPFBLNZXDBXWOAZDKXXNHGPEBQBMSQGPDKLEWLWWTOOXC
HWIUZIAMVHXYFTMFNCMEBMCYCTJASWVHIDIDKAQLYSRSEDOCZUSTKQHXDIMMWEQADSRLUOLSFWVGSSWKCXSZDLQHI
XYKABRPTGFWKDNNCVGTYPDDTNJHCWRPHWTBZYSMBGBOJSIKUMYLYOJZTPKZTJPFRDSAUXHXMEUDACXFLCCLPDQPCX
HMQYRXQYIXNKANSHYITIMBBLRWALSJAWFAHIXNURFYXMHHHZNLTMJTKMYITIHKPUPYNIRBYUMJJAWTPUUBNJJUHMK
HVNIVDTHZVBAFSAIMSHSZJFRBHZZPMTYOLOXABQISAIJFQPGFKOJEEULPUO

String2:

TRHMBIFMAKCFCZZZWJXAPIZGDZJGRSLOWAOICNVIQPZMMIQYPXFRHSOHBWQDLTWIQPRPUNQWJAQTYDNXAVONLWQBG
WOZSMPIJSYLXBMGNTLQETAOUXRARLFXHFXTVOIJCVVYXBJDBDIBZWCJFQXORHOMZEKOAUDWLHHGSPIQYAISBMEIRL
KVTZLZVRKZOUNZGLMZIALILEUZGGUHFWNUGJMGIOMWZLZXAXPWTJJGRZCVZTHSUYQQMTZTMRNVEIHMQKWAIAHSPLP
CIRYRBPLIGSIRUQIILQODZBPDDQDJOZFGQNFSYGZMYRZJEQZPFCXRSUBTFCWHFRQUCIHVOLSICCDSLLJRXMALKWHO
MHOJXTTEGFKAXNZKRBXHJJCRFJBGYQVILMYYRNSIWHWBZPUHLIZARIQORBBGLWZUYEXIKHSNCBRGLNRAZAJUSCOYR
PBFQABVSXNZGOONYUANDDBZVXXGFIJPIGZHIOVUMLIAXMRNPUXLJLHIVBVSHXCXWVXWYBMMLRDBXGKXBJMSTFFAGW
WPXAUUGMHCMIKQDUYESETQYSPFNZGUMYHQLHAXQUMUDUERQZRYTYHARVBSHNVHUSFEREDVTCUQKXCLIJIKRQYMNBW
BRBZXTUHZVJIAPBBEPRRMRBRGPMGUAAIVKHMBIPCTZZZZUIJAEEETZCYBBISLHVTBBXZZFNYLUDFCTDQGLISFEGQH
…….DUUIZCDVQXHOGGVEUGAWKSKMPHMZKMUBUKFFIJMDBYIPOOVQGNFQNSYSWBXCXQM
IYPZIGRNFHESHMMUIWTLGJKDWDAPQXHDUWOOIVBWFPJGSARBTBHDNVTTDUHAGAQDMWSATSIUUWMQXWRCFWTZEBFKB
WBWRUEUSXYSVFEGQLSDZQAXDBFZTRWWLGIVHKXERKYOZUEOAMFKHUDIHZKVPJRKKLIBFGLPFMEDKUSVUERIQBZWYU
NAKVQWKDWCNLVMKYVBLBQKZCTXNFWLIKSEHIPOQJTGOPFMIOZMNTTMBPETPAEDNBTQIQJFETNUQQTNXBVZMAKJXIZ
AOSUVLBBESVXLKFMUQMGKLQQNUUTRMBENCNOLCNDFQDRIGFVCDCISMSKSYXSZWXBOEWEOOBUUNUVPTXLXUJPOQAUD
FHKDKPFVENHEDAKBEZYCPWFYKFBMJAUVMHNHZCDJKEGQFFBTAYRRRMFWUMASVRXMZRPKFRYNPRPVCKUPSFIVLWLTR
KAKUURXPZHEMVYQTWUOHGWUBARWYZQQMRYDKGJNHVWZBMXGVOWFSNIMGHLEOQPASOSZWUIBIDWBHJYSEFBOKLWQUV
OTQSYTPTQGYYQMUEROUVAHPBJAZCEFOMEMGVPSHEQBHRYIFULTYCJVRWHLGKHHOAIAWGGMHDHGTIWRALHLXRHVHSI
HDAQZTVWGGYLTDEIXBICAETKKOWVECWSZFITGTJSQIOGWTMBDHQBAQIUEBWAXBUTVBDAKJMMCDXYCCVNUCOUYZFNB
VTYMZKONHGGGDJMKVNQVSXOSTXWVZOKTIDCYNORFNCUHUIKPZNVQQWWYAKHTUEAJBJHLUEFJPYQLPDHMAOTSTEEMJ
GRYZMXVMFVSREGMPAJVPAXYOJCIOWJMEEKNTRTVPCUWTMAFJDMYXXQNPLYIKEHOJDGOPFQYGKFDYNYSZGNFIXXXJM
CDQYMBFEHRRXLWXOBFBPTBXOKUFWUQQHLGVQBQRVMEOXZSWHWKMGBCBYQGPQJEEZYLYSMWNNFAMNSKVPFWAHAJMIY
INHMWBJHHQSHHMDVUIJMGKPPBXPHHPJYCINRGPOSNQOTCJVNXOKKRLKOVQCERGKBHBSSYSKXXTLHCMOFLHUKUEJMY
EXMYSQABDVBNKYGOXTVNSKWJIYIIDTGIQA

LCS:

NSCDKQYKMIESGEEOHWDFMNWHPHOYWTJHMRMHWPPLWUZOKBIBXCVPQKYHLXANLYUXKRRNWVXFANCDDNCSCWIFIQUZY
FUPSGRKFMPFGLBJIZIDKYCOATOERAXMFZKSMMHBLHYZOSNKQMYSKCQKGIQKJHWQZTEAELTMSJPXUKSPTTWYKAVKGE
MVFPNCYWENLXOTPAMJZXHFJWTLYIRPDDYDEGDIBFJDWQODGALVNKTRUWKENJAJGLRCJNDZAKOMIJAJWJPBPURKBNI
NMDBRXDHWHTEHSNBAMHPXAKCJAPVPNFMJCFDXSMMLUTHVJAQXQHNZNYMRDFLXOZVOYAIEAUSBRRMOSOVSJVXGDZMJ
……DUYNBLBAITOUFXSUDPBTRNGWEIHIWJPYTHDBOSGYGRWHGXPPWZMYTHISGEZKYLTVWUXROBCXACCRZJUX
VUNVRUQPFQUVGVJCARXPYCKPPFAXVQZNHNYJHCUDEQIEUXTSNCNLIXJGBTIBISFSYBXWOTGH

4> RecursionMemo:

```
/* Objective:
 * 1>.To observe empirically complexities of different implementations of
algorithms
 * for the same problem: finding longest common subsequence in two sequences.
 * 2>.To find out how accurate are the theoretical estimates of complexity
 * when compared to practical execution times.
 *
 * Version: 2.0
 * Author: Aishwary Pramanik (ap9599@g.rit.edu)
 */
import java.util.Hashtable;

public class RecursionMemo {

	// To store the state defined values
```

```
	Hashtable<StringBuffer,StringBuffer> storeLCS=new
Hashtable<StringBuffer,StringBuffer>();

	// Function to find LCS
	public StringBuffer findLCS(StringBuffer A,StringBuffer B)
	{
		// To store Substring
		StringBuffer sub1=new StringBuffer();
		StringBuffer sub2=new StringBuffer();
		StringBuffer sub=new StringBuffer();

		// Consider the base case
		if(A.length()==0 || B.length()==0)
			return new StringBuffer("");

		// Consider the base case
		if(storeLCS.containsKey(A.toString()+":"+B.toString()))
		{
			return storeLCS.get(A.toString()+":"+B.toString());
		}

		// Characters are equal
		if(A.charAt(A.length()-1)==B.charAt(B.length()-1))
		{
			sub=findLCS(new StringBuffer(A.substring(0, A.length()-
1)),new StringBuffer(B.substring(0, B.length()-
1))).append(A.charAt(A.length()-1));

			if(!storeLCS.containsKey(A.toString()+":"+B.toString()))
				storeLCS.put(new
StringBuffer(A.toString()+":"+B.toString()), sub);

			return sub;
		}
		// Characters are not equal
		else
		{
			sub1=findLCS(new StringBuffer(A.substring(0, A.length()-
1)),B);

			if(!storeLCS.containsKey(new
StringBuffer(A.toString()+":"+B.toString())))
				storeLCS.put(new
StringBuffer(A.toString()+":"+B.toString()), sub1);

			sub2=findLCS(A,new StringBuffer(B.substring(0, B.length()-
1)));

			if(!storeLCS.containsKey(new
StringBuffer(A.toString()+":"+B.toString())))
				storeLCS.put(new
StringBuffer(A.toString()+":"+B.toString()), sub2);

			if(sub1.length()>=sub2.length())
				return sub1;
			else
```

```
                return sub2;
            }
        }

}
```

OUTPUT

CPU Time(ms)	Length of String1	Length of String2	LCS Length
0	1	1	1
0	2	2	1
0	3	3	0
0	4	4	4
0	5	5	3
0	6	6	4
0	7	7	2
0	8	8	6
0	9	9	2
0	10	10	6
1	11	11	8
0	12	12	2
0	13	13	10
0	14	14	10
0	15	15	3
0	16	16	13
0	17	17	13
0	18	18	12
0	19	19	6
0	20	20	16
1	21	21	16
0	22	22	5
0	23	23	18
0	24	24	7
1	25	25	17
0	26	26	7
0	27	27	21
0	28	28	7
1	29	29	24
0	30	30	23
0	31	31	8
1	32	32	25
0	33	33	24

1	34	34	8
0	35	35	11
0	36	36	9
1	37	37	11
0	38	38	10
1	39	39	8
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
10753	62349	62349	16389

String1:

GRUVUPLJLGVRZQBGGPJJGJMOAQOBNJJZYYEMWGYIEQQJWOOWXFDYBOZDYEGPYQBMXWTBUZMDXI
IVSPEHUVKKNUCRXYOBLXDNBTNCLDTPDFPCVHXXXIGHGREUFQDQKKRLPURRLIFMXMBGJWKQFHKT
ATHFRIIDCGKWNEGPGFWCJJPUTEXEEGRXRBNGAEMCVWAUJPIYYPMRQADMMJUXTRWHBMHHYHNHUA
PLGUGUUUWIXEVNIZEXPZKBBUVEIXHZIUGCARROCOAUFKAMQOJAVPBWFRLNMJOZJAZZALMIITOL
JGPZYIZLCVILJJGUCWXIHXLVCMVZSGNXQTHZLHLQBTGAVFEPVMUTRWUYMREUYTFQSZOEZVVECG
TXHWRIAVRULNSCVNJQTWSAUXXGPORLYBYLAMHLLSTLRIHVLKDLIRDUJGNCGTNVQTUMGFXOEAVH
RQZCTZWKTWSPZQMBJOAVYSNWFGEXPQGRBVFXLCYLNXRDDZTCPLPFRCGMZUHDCOFYDFMRQVAFFR
NJJAZYKXJIHSIOULOKDYOYVLYHZLLXTOWDISRWNHMVCZNLWHLZFLYIKYFDADIJCYAAYJLLWLZO
RIQZIGQGGQSTOPHFYKABRQGSZMBORFPZRTWBWTIEBVSOXTWRSUTGOOCMCAKIGLMMKWHPSFWTDM
OLOOPNJJOMDOGEJTYCTKGFZDCGGWLTHFXKYEYBKVDHSWZWBTVDFNXYVFAZSQPGMIHUTXUFIOHQ
PYTCWDLVKYTTNBLPNUQZPFGWMRXNZEJVBCOEBAHREFLCEWSJNNWHEVUFTLTTAWTFWPKGBKODDT
IDTNCEKRPUAPNOFRXUVMUKQXUAMUHEJVIPROGBCGAEETFSOVJBKLDPNCGGVBQCBRJXQVAJPWBU
MFYYNGLKZYNOAGKUZPWQXHRLTGHPIACFQYGCRNZIRNMOPSNCZQKDVBYANGIXNFGLLNXEAQVKNV
BKDTNQOPPWLCVAKOLQJDJNJFTTOREGIARFBRCAKXRILDQDWPVQVWAKJIEZCMBRDWWWNRXJKGQU
TQIYMHPMKTREMSEVHCQNHDDGRBMTINYRYNLZUKJWISAACOHRBURAXLXAIBBVOWVASRXXDEWPCZ
MGELPQZSKWHIOBSYRYKYNOTOCYKDBLYVTLLNRUAPJLEDTMBFQMEQWCTIGAJRESFXFFADBUUDHN
RPLABAIOBMJXUQUPLLODIRKRGBFPMSKMBKBOEHZXPXUNBZHXVZMSLZMLKXSMKLQCFXBCVJMHNN
LBIDNYQGFPJKATRVJCZQSXDDPZMCVEGKFLNOGYBXJFJFBDVHRAFUIHILFXEOWCEJLGIXIYLUTJ
ZIZVFLKDYXJEXGKJSIAXGLSULHROVOMDDNCJITFQRXRONNILFTLDCWPCLWSNXMWVRJWNUFHYWJ
QOBZTWYZOUPNMRACAXQKMOTGFNCRGSNNKKFQKTPGDYFRNOIEZYLQQGGUGJUEZPYGVRSCIZTAIP
IQBDTICBUFNVUMFMNAPRLRUVHXEPAIRYBMVYFGYSKLAAZOSITABFISEXBJLIEBEAQQQCDFAUJR
ABXEKEAFBVKFBZRYLDUJATHTT……QPWVNPXVAGQFOTFELBPZJSAGVAYVTDCOXHSEKRZTAJJP
MZPDMTEPKCQILGAKOSSKPXSKAVJEYJAMMJQCWLXZWBRJVENKWKBZIEMFPCFXWJHOXLRKDRTCWT
BDXFEEFRWDLXGPOWZUQIIQMAWSBWIFRMGSZIHJSDYUUQXIZJELAYEXUMTSKUOSKGJLPOWDXYRR
TXIIBWDRZROKSPMLEUSUECFEUAQJRWFWTRGTWCPUYPYDVORWRHPKOWWFJHUBCFPBKWPVVTHMDX
RSQZOGTHVIZXQSUKWKWCJDHXVWJQIOXIQFNVJTEHVLIRUHGXKPKRDKMTHSDFSUQYJVHTKVNIRK
IQYPNHNYRYEWNCIFATEWUDQSOPZVFZFNBDLRZVYFILTIGVCLLYAGISACHLSNSVRLSMGNXFZKKY
MUCVKYZRATIHMIGSFXPWLZEMQVJGDDMIBAHTJHZRGNTBIGWEBZYZZIMYKVEKLYLZPXIOPEQSJE
VHISUFUNHIDEEZRDKIFMPJDHKRYBPUMVENPOPLUOTTRDBOCWZIIJCWROHVFXLEPUYWATFYMROH
BVZJWIJAENXGFVYEFFQGBJFLKYMWRMBHMTDXATEXFJNPSGYNYPZTSHRVLEBFZDCLGNAUSGYFRU
RQUXFPWOAOOQAXZDFUKWZOENHQTLWVGWXJJCJYIVQWGWSMAANPNHIJNIDBOHGHKOYMEHIBJBXC

……..OFGZYCZMFHZCYJKIIJXNUABQUQRBIHQVIHAMEKWTVUSBBSPZVDTTGFNONBFGWREHUOYGH
BIYXQALECWSIYYOMBPQZCUWIAZTXKVRSXMOGMCBUZXVFUQQPNQIGOGPOWMYIDFTVZPLLAVREBV
WZNPVBMARDNPNONZLGYIWBGIWBGIPADEPWSNXDYZYPNLEGIGIUBLIGRTEMYMXICKEUNLUVUCJD
WCKXBAYLUVMBFIMMQTGSYZFQGANIHQAFGNCUBWSUDKDQEOEOADMYPDBUNOTMCBKLYTKNJWDDEC
GTFSRWEJVGXOMIUJCTGGZEOHHCZENHADRAXAESVXGAFXSLOQOJHUNDGCBAYNYKZQKVHAFPDRZO
SBQSXOLMQHYLYTCGZGBFJYZUWVBJMNYQQAFOBHFULBURSIRYLSEMGLZRQABFFOHTJCNKIZTFSR
FMKNTXFPNWUNLOYEDMWVBQPJPIKKKZTMRVZRLGGFWAZXMVUXOHKSZDDBEVSVROCSYWZPALPZVJ
BQYQWBXQBATLSYTFQFWQGCYVWBHTWWARTURBWDUDRBFISLHDBEVFXSFTWXFCGDFSVRPFAKIZUL
HTTNWKSGCWXUPWZERVOOGZJXQENCGRLXHCQCOTGJYREWUHQUYQJHVJMDFXPYUYIGJIJWSGZACR
LOQSNVDAGNGUQKPGBBPDYMOVZIZWKEXIYCHQAWTRLEVYEIJYINOIPDHHHXQYNTFRUKYJPALXFX
MGA

String2:

ZRHKMXJZOYSLWENZQMQYMFQMHYONJVVWFTCUZZNRVDAIZLLHJUDRSWYRHSAVUOYMXPARDXLVEK
THAMQZGACOXEQDZXJMLGDNEMHUSWKBAADVYVDIOKDUDKBADIPYULFRKDVKSXIPUBABEFFNTKOZ
HZKMGLOUSHUDXBYEAIEXYVWWETXUUZBAGNOTSTZMBZDSEEYTHFQREHEJJHKVUHYQMFONVWCLOS
KRUCDTFVDXTRQQLHGWKKDZZUPDWKGWEWYIPJJDTTJIZVREYJBHRYAQUWARQFNBBUIMLUSMKEHM
CBVEWTUNWIVBWSOTJTYBIPNHRFEFGVUVTYPUZVYKJDGEEDXVLFHNVWFRFCQXFMBYFETJHXEUQC
PQSTOJNMVCFAYELYUOORROORSFPFCDNFRWGVJWNKNJOZUMSXPZWHVRSMFMSJHAMJQIJMBQNDXI
SVLLPTHREFYUATCBNZQFBXVACQGREPWYBVQIQPBZSEHRYBSBAPWKAHLYHIBNBCDSVFIFJXTYSB
FSREQMVBLFSJKNWVBMIHUJEJKKHNJMTSUWBEQIWVMKVOVORCXLVMNYIJJEYAFUOOVLQLYBKTHD
HPLNELOMHSQKRIMAVMXJJASKBDFSLHPYGFGAXNXHSPPGTLHEKPPOCKCGBDASFAPATPGSTBZMYZ
BMNPNPRJFYKJPWMKHFGYTVTEBXBZXZWKBZVGWXTGBFUKUROIVHTNLFKJZLRQWCJITGCTGNEFNI
XNFJUEXBOAKJNWKOJPJQPHKVJJQHNHPMNZWXGPVZXCENCZEBPWJVWPRHGKASBKBKCNULLYEFRE
LMRNGCHSFJ……YBXJGZJOVGAOJJXGXYMPPGNWBNLETCCLFGJJKPHMMVOQIGNYMBFQUCJUTMD
YUIUBTTMOSBABAMSGCHEQRWESHAQYQMFHJRGKKOCHXIIIITJEZIKLQQUHADMJUDSPYWCSAODNV
GVDKSIQVATKJSXFJDNJKVZVGYKBLELYPWBEYBNCDDUTRUWMHYCNZWSAWULLCASMAEAPZERFGOF
QXHDKIHTROPJSITVIILVQWBZEMLLDTAWDWFRJYZTTNVXCLCCORPYBYQAEISNPVEJBABZIOLRWY
SIIQQHZBZEALAZJEBFKELGFXPDGVNBULDPURNPJEHUDTKCYHFSIQQRXXYWHCENEJPVCZQYWQAD
OFDZZCQUWHAPXPOBRGKSVFSMSLJTRIUIIUWBSVRMDTSCSQXRNXWUVONPOBWAEHNHPAAIQGGRFD
RWWPXHOLMRKWYVKWYTXBTROKOYVAUZMAYPBEQJDFPJQPRRUJTMBXMDIBLDSWFPFMQRDNKSUVAQ
QNSOOHHQEOUQVVUVWOTLFSLJSGRUZDUWXEYHTVFLWRBVQVFMOJSZQCSNYZZJZBQJTJHXZDCBZO
TUDYDBNEHPM

LCS:

RJLZQMQYMYJWFDZDYYMXDXVEHCXLDNBDDIUDKPULFKKIPUBAETHMLUUXEIEXVEXZGOMBFRJJMO
CLUCVXQLHGPWEYTZVEHWARULSCVNWSOYBHRVUVTUGXVHRCQMBYFEXPQVFLYRDFRGMZHFMAJJXI
SVLLTRCNZFACYIQZSHYBSBPWBBVXTRSUKKHSWMOLNJJOOYTDLHKVSBDFYFASPGTHPCDTTNPNPR
JFJWHFTTWTGBKOITNKRQJIGCGEEOJKPJQVJPMNGZNZPWHRLGHFYGZIPNCZKBYAIFAVNBQWLOJD
NECWVVBWXUQIHRCTYYNAAABBAECMEQHRYKYCYBBFQIESADUUDNPBAJUUOBKKZXPHLZLSMLCCHQ
GFPKXMVNBXJBVRAFOELGIXYJZVLKXJGJSAROVDTFQRROITDWPSXMFJZZARSFPGDYNEYPYSZAPQ
ICBFNMMPLHXEYVYGLSIAISEIBQCFURAFZYLDTHTMBDZUAOKZLSYSKXSYVNVHPUUQCZPYMMNKFI
WPQFTFELPSVHRQIGOKPSKEAJZBWZEMPWJHRBXFEWLXOZUQIQMAWWMZJXEUMSUGDRZKMUEEGWDO
RPOJUBFHDZZSUKWVOEVUGSUTIKIYPNHYEFAUDQOPRZVTACHNRNYUZMSXPLQGITJRGNBWYSEVEZ
DDYBPUENPDBZJVLPWWNXFJWBXJPPTSVEZXFOQAWZENTWWXCVQSAPHJNBYJBXCUXPSAYKEFLDMQ
GRWEJBJQVBFAJQDMREEMINDNSOOCLLJKP……QSHJEVBJWRHBYKZZWTHIJJZGKAMJVFGXVQOGIC
IFZRIKXMUALNWMOSTAYHRTGGXAWDSKLWMMINRYQNZQIWMQIHEEKMMYDIIZKJWNKJGHVTSVUGOH
QWBKNLGUUKRKJMIUHGUSTNFUXLEQWFUWYRVOTPNORSYVDRXSJHTTNGIIZKWBNGRZRUICKOWKRJ
VVEDZMUDULNBVRHEIXGAEKCGMTCQNBYNPMOGLBUCDJAJDEPXXUYQMWTXNCDRGCVDMFHXFMMQGK
ZIIEKSUVSTNUSUCLFDKAGDJYDCUVUVWNOQVCZUKZMKUXCBCHUYUYWYGJZPMVGQXEVAVVZVYQAX
RPMVAAXLAOZZMFZYIIXNQUBHVAKBBSPVOBRGIQAESIOXKRCVFQQQOWMFZLLEZADOZLYBGGAXYP
NLELGMMINUCJDYUMBMSGHQAFGCKQADMYDNKTKNJWDDTRWMHCZAAAEAFOQHDKHROSBLLTFJYZVB
YQABBIRYSELZBFFHTCFSRXWNEJPVZWAZUXOKSBSVRCSWPAPQWXLWYVWBTAUBDUDBLDSWFFRNKS
UVOOQEQOTGRUUYQVMJSZCSNYZZQTJHXTUYPM

5> lsqtLCS:

```
/* Objective:
 * 1>.To observe empirically complexities of different implementations of
algorithms
 * for the same problem: finding longest common subsequence in two sequences.
 * 2>.To find out how accurate are the theoretical estimates of complexity
 * when compared to practical execution times.
 *
 * Version: 2.0
 * Author: Aishwary Pramanik (ap9599@g.rit.edu)
 */

import java.util.Arrays;

public class lsqtLCS {


	public StringBuffer findLCS(StringBuffer A,StringBuffer B)
	{
		// To store the proper LCS
		StringBuffer sub1=new StringBuffer();
		StringBuffer sub2=new StringBuffer();
		sub1=findLCSHelper(A,B);
		sub2=findLCSHelper(B,A);

		// To consider the appropriate result
		if(sub1.length()>=sub2.length())
			return sub1;
		else
			return sub2;
	}

	StringBuffer findLCSHelper(StringBuffer A,StringBuffer B){


		StringBuffer sub=new StringBuffer();
		int row1[],row2[];
		int i,j;
		int m,n;
		int fl=0;
		int loc=0;
		m=A.length()+1;
		n=B.length()+1;

		// To arrays to save memory
		row1=new int[n];
		row2=new int[n];

		// Initializing the arrays
		Arrays.fill(row1, 0);
		i=1;
		j=1;
		// Looping through to generate LCS
		while(i<m)
```

```
		{
			fl=0;
			j=1;
			while(j<n)
			{

				row2[0]=0;
				// If characters are equal
				if(A.charAt(i-1)==B.charAt(j-1))
				{

					row2[j]=row1[j-1]+1;
					if(j>loc && fl==0)
					{
						loc=j;
						sub.append(A.charAt(i-1));
						fl=1;
					}

				}
				// If characters are not equal
				else
				{

					row2[j]=Math.max(row1[j], row2[j-1]);
				}
				j++;
			}
			// Reinitializing the rows
			System.arraycopy(row2, 0, row1, 0, n);
			Arrays.fill(row2, 0);
			i++;
		}
		return sub;
	}

}
```

OUTPUT

CPU Time(ms)	Length of String1	Length of String2	LCS Length
0	1	1	1
0	2	2	1
0	3	3	2
0	4	4	3
0	5	5	1
0	6	6	1
1	7	7	2
0	8	8	6
0	9	9	1
0	10	10	5
0	11	11	8

0	12	12	6
0	13	13	3
0	14	14	7
0	15	15	9
0	16	16	4
1	17	17	14
0	18	18	12
0	19	19	4
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
4390	5352	5352	223
7740	5353	5353	2698
5730	5354	5354	208
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
11376	137982	137982	54408

String1:

1000100110011111111010100010100000000100111101100011010110011101101000011111000010000111111001000000
1111100011000010110001000111110111110111101001010011000110001101001000100010000010001100000000011
0101101011111000100000001100000101001011100100100110001011001011010110001101010000110010110001010
1100100001100011111001001111011111110111011010101110111011101100101000000010101000011111110100111
1100101000000111001100111000010000101111110011111111011010100101101110000101100110111000101100001
0001010110101111010011100000110010001110111011110010010100001111001001010000010111000010010101111
1111000010000000011010001011011000101101010100011010100111010110001001010011110001110111000000111
1101001000011000001011001010111000100101111101101110110110100111000101010011101001100001011111010
0110111011011111010111001001001110111101110000101111111010011101011111110010110000000110110101111
0100001100101101110101100111100101110010111111001011100000100011110111101100100001101001111010111
1101011011011011110110111111011000111100101100110101011101010100010110100101111001000011111111001
0100000110001000011000110001101010110110010111100111001011100101011111100001111011011111111011000
0001100111000000100100010010111001011111001001010000000000110110100101011000101100011001110001101 0
0011001111111110101010100000100010110101100001000010011101101101001010110111011110110011111000111
1111101100101110101010010001111100110101100010101111100011000100101111110011001100010100111010101
0001011100110100110001011010000000101100100001001101011010101111011000111000110001011001001011001 0
0110101011110100100111101001001110011010001000100111010001100000101000011000111110010000001010011
101111101001000000000100110110…………1110011011010100001011011100110101101101011001001010010111110011
1010010111010101101110101101111001001111001110110010110100001000110111111011011100111111001001010
1100110001001100111010000000110001011010111100000001001001101000111110110110111001000000111111111100
1000001100111111100111001010100011110111110101100000101001100110010000000010010011100010101010111101
0001100000010001100111000000001100001001110010000000100111100011010101000111011111110001100111110111 0
1110011110010011110000010101000000110101100000001101000011111011001011010100010000011100011101011 00
1011101011110000011110110010111100011000101010100111101011001110100000110110100100011011010101100
0111100110011111010001011001001011111010010100000110101010011011111110010010110011101001011100000 1

1100000010101001011011110001110011001000100100111110101001010001100001100011101011001110011100101
1101001010101100011000100010000010101000100110000111100000111010000101110011100100001000100010010
0011100000011000000011011011110110110110100100110110001010101010100011111100110001001100100101111
0110011101011010011010111001010000101101010001110101010001100100010101100111101111110101000001111
1111100010100010110000000011101100101011001011011110001111101011111010011110011001001100010101100
10100
String2:
1100000011001111011011001010011001011011010100011011111110100110101000110000011110101101111101001
0010010001100111001011110011111000101011110100100001011100010101101100100110001100101010100111000
1001111010110101011110011111100100011111011001101001000010111011101101110010110101101000000000001100
1100001111001101000101101000000111011100110011001000111000100111000001101101011101111000010000001 0
1110000110100000110010111101100101011111011101001001000011010000101100111101110000000110111110001 1
0001101111101010001110101010001100101010010000101010000110110011001010011001110011000000101111010
0001110101100000100001011001010000110100011111001000110001000100011111111000110000000100011011011 0
1100100001101110100100000101011101000111011101000000101100001100011111001000101011110000111101110 1
0111001101101011011010111010011010000011011111100000010101111101100110011100001110001010000011 0
0100100011110100000011011011010101111100011110001100101000101001100001110110001111111101100010 10
0001001110100100111001110010011011001110100010010000011110110110111100000100011000110100111101 10
0011111001100000001001111110001101100000011011100001010001001000000001001111100010100110100100101
0000100100100001010001100000111101001001010010011110100000101110001001011001100111101101000100 10
1100000101010101110111100110100110101111100000100101101010001101101101010010111011011100001111 0
0011101100101110101101011010010011000010011100000011000110100011111111111001011010100100000101110
1001001000011000110110100000000000100111010011101001000100001011100101111100111001010110111011111 00
0010101110010100100000010111111011110111111000001000011111111010010010100111000011101101100110101 01
0000011000000111000000111101111000011110100110110101001101001110000110000110000010111111011001111
1100000010000010111000111001111000101111010011101010011010011100001100001100000101111111011001111
0111110000001010101100101010011111100001110010110001001110011101000100101100111001010110111001101 0000011
1011010111000101011101100000001100010100000010101100101010001100110000101101101111110110101110 1
0001001101010110010111000101000110110001101000011001101111101111011010111000010100110011000000000110
1001010110011001111001110111100111101001000010000111101100001110001111111111010101100000000001001
1001100011001100110011001110111100100011100110000001011110110000011111110000000011111000111011100
0110100011100110001
LCS:
1100000011001111011011001010011001011011010100011011111110100110101000110000011110101101111101001
0010010001100111001011110011111000101011110100100001011100010101101100100110001100101010100111000
1001111010110101011110011111100100011111011001101001000010111011101101110010110101101000000000001100
1100001111001101000101101000000111011100110011001000111000100111000001101101011101111000010000001 0
1110000110100000110010111101100101011111011101001001000011010000101100111101110000000110111110001 1
0001101111101010001110101010001100101010010000101010000110110011001010011001110011000000101111010
0001110101100000100001011001010000110100011111001000110001000100011111111000110000000100011011011 0
1100100001101110100100000101011101000111011101000000101100001110001111111111010101100000000001001
0011110011111100011110010110010101011100010011010101100101100010110110000100010101011111001001010 1
1101001100110101011011001010101111101010011101001110101011101010010001100101111110110101111101010
1101111100100101101000000110101010 0 1010100101110011101101001101111000111011101001111011010111001
0111010111000001111101110010001010111011100111111011111011111111001110000010001100001101101001011 0
0111010111010110000000000010011011000110000001101011010101111100100101000101010000000010011111101 0
1110100100001101111110001000011000000010010001011110010011101100111110011010101110100010011011010
1000001110001000101101101010101010011010110011100001001101110111001110010011000010000110001011101
1110010010100001000110001101001111011000111110011000000010011111100011011000000110111000010100010
0100000000100111110001010011010010010100001001001000010100011000001111010010010100100111101000001
0111000100101100110011111011010001000101100000101010101110111110011010011010111110000010010110101 00
0110110110101010001011101101110000011110001110110010111010110110110100100110000100111000000110001101
0001111111111110010110101001000001011101000