Universidad del valle de Guatemala
Algoritmos y estructura de datos
Delbert Custodio, 14246
Yasmin Valdez, 14079

Profilers

Resultado:

La implementación más efectiva según la evaluación con profiler del tiempo de ejecución para calcular sí o no el conjunto de desarrolladores Java es un subconjunto de desarrolladores web fue el **HashSet**.

HashSet (48.752 ms)

Hot Spots - Method	Self Time [%]	Self Time	Total Time	Invocations
java.io.FileInputStream.**read** (byte[], int, int)		48,752 ms (99.9%)	48,752 ms	59
java.util.regex.Pattern$CharPropertyNames.**<clinit>**		2.7 ms (0%)	3.27 ms	1
java.lang.invoke.MethodHandleImpl.**<clinit>**		1.40 ms (0%)	2.17 ms	1
java.lang.ClassLoader.**findBootstrapClassOrNull** (String)		1.2 ms (0%)	1.3 ms	25
java.io.FileOutputStream.**access$000** (java.io.FileOutputStream)		0.980 ms (0%)	0.980 ms	1
java.io.BufferedInputStream.**read** (byte[], int, int)		0.915 ms (0%)	48,753 ms	63
sun.util.locale.provider.NumberFormatProviderImpl.**getInstance** (java.util.Locale, int)		0.841 ms (0%)	17.4 ms	1
java.lang.Class.**privateGetDeclaredConstructors** (boolean)		0.797 ms (0%)	0.817 ms	6
java.lang.invoke.MethodHandle.**<clinit>**		0.664 ms (0%)	3.63 ms	1
java.lang.ClassLoader.**defineClass** (String, byte[], int, int, java.security.ProtectionDomain)		0.656 ms (0%)	1.4 ms	7
java.io.File.**exists** ()		0.515 ms (0%)	0.555 ms	10
java.util.regex.Pattern.**compile** ()		0.502 ms (0%)	9.1 ms	4
java.util.Scanner.**<clinit>**		0.486 ms (0%)	6.12 ms	1
java.util.jar.JarFile.**initializeVerifier** ()		0.479 ms (0%)	0.899 ms	1
java.util.regex.Pattern.**group0** ()		0.478 ms (0%)	2.81 ms	54
java.io.WinNTFileSystem.**canonicalize** (String)		0.460 ms (0%)	0.704 ms	4
java.lang.ProcessEnvironment.**<clinit>**		0.445 ms (0%)	2.33 ms	1
java.text.DecimalFormat.**<init>** (String, java.text.DecimalFormatSymbols)		0.400 ms (0%)	0.977 ms	1
java.util.regex.Pattern.**expr** (java.util.regex.Pattern.Node)		0.387 ms (0%)	7.43 ms	39
java.text.DecimalFormatSymbols.**initialize** (java.util.Locale)		0.351 ms (0%)	8.4 ms	2
java.util.regex.Pattern.**createGroup** (boolean)		0.348 ms (0%)	0.391 ms	35
java.util.regex.Pattern.**closure** (java.util.regex.Pattern.Node)		0.331 ms (0%)	0.367 ms	82

TreeSet (46.945 ms)

Hot Spots - Method	Self Time [%]	Self Time	Total Time	Invocations
java.io.FileInputStream.**read** (byte[], int, int)		56,945 ms (99.9%)	56,945 ms	59
java.util.regex.Pattern$CharPropertyNames.**<clinit>**		2.33 ms (0%)	3.65 ms	1
java.lang.invoke.MethodHandleImpl.**<clinit>**		1.36 ms (0%)	2.11 ms	1
java.io.BufferedInputStream.**read** (byte[], int, int)		1.11 ms (0%)	56,946 ms	63
java.lang.ClassLoader.**findBootstrapClassOrNull** (String)		0.938 ms (0%)	0.943 ms	25
java.io.FileOutputStream.**access$000** (java.io.FileOutputStream)		0.846 ms (0%)	0.846 ms	1
sun.util.locale.provider.NumberFormatProviderImpl.**getInstance** (java.util.Locale, int)		0.780 ms (0%)	17.1 ms	1
java.lang.Class.**privateGetDeclaredConstructors** (boolean)		0.771 ms (0%)	0.789 ms	6
java.lang.ClassLoader.**defineClass** (String, byte[], int, int, java.security.ProtectionDomain)		0.666 ms (0%)	1.13 ms	7
java.lang.invoke.MethodHandle.**<clinit>**		0.657 ms (0%)	3.56 ms	1
java.io.File.**exists** ()		0.618 ms (0%)	0.647 ms	10
java.util.regex.Pattern.**compile** ()		0.606 ms (0%)	9.80 ms	4
java.lang.ProcessEnvironment.**<clinit>**		0.561 ms (0%)	2.38 ms	1
java.util.jar.JarFile.**initializeVerifier** ()		0.531 ms (0%)	0.961 ms	1
java.util.Scanner.**<clinit>**		0.495 ms (0%)	7.2 ms	1
java.util.regex.Pattern.**group0** ()		0.469 ms (0%)	2.74 ms	54
java.io.WinNTFileSystem.**canonicalize** (String)		0.460 ms (0%)	0.680 ms	4
java.util.regex.Pattern.**expr** (java.util.regex.Pattern.Node)		0.385 ms (0%)	8.1 ms	39
java.util.regex.Pattern.**RemoveQEQuoting** ()		0.377 ms (0%)	0.478 ms	4
java.text.DecimalFormat.**<init>** (String, java.text.DecimalFormatSymbols)		0.372 ms (0%)	0.965 ms	1
sun.nio.cs.UTF_8$Encoder.**encodeArrayLoop** (java.nio.CharBuffer, java.nio.ByteBuffer)		0.355 ms (0%)	0.455 ms	107
java.util.regex.Pattern.**<clinit>**		0.353 ms (0%)	0.400 ms	1

Universidad del valle de Guatemala
Algoritmos y estructura de datos
Delbert Custodio, 14246
Yasmin Valdez, 14079

LinkedHashSet (52.269 ms)

Hot Spots - Method	Self Time [%]	Self Time	Total Time	Invocations
java.io.FileInputStream.**read** (byte[], int, int)		52,269 ms (99.9%)	52,269 ms	59
java.util.regex.Pattern$CharPropertyNames.**<clinit>**		1.94 ms (0%)	3.29 ms	1
java.lang.invoke.MethodHandleImpl.**<clinit>**		1.27 ms (0%)	1.97 ms	1
java.io.BufferedInputStream.**read** (byte[], int, int)		1.4 ms (0%)	52,270 ms	64
java.io.FileOutputStream.**access$000** (java.io.FileOutputStream)		0.966 ms (0%)	0.966 ms	1
java.lang.ClassLoader.**findBootstrapClassOrNull** (String)		0.916 ms (0%)	0.920 ms	25
java.lang.Class.**privateGetDeclaredConstructors** (boolean)		0.753 ms (0%)	0.773 ms	6
sun.util.locale.provider.NumberFormatProviderImpl.**getInstance** (java.util.Locale, int)		0.752 ms (0%)	16.3 ms	1
java.lang.ClassLoader.**defineClass** (String, byte[], int, int, java.security.ProtectionDomain)		0.610 ms (0%)	1.4 ms	7
java.lang.invoke.MethodHandle.**<clinit>**		0.599 ms (0%)	3.33 ms	1
java.util.regex.Pattern.**compile** ()		0.555 ms (0%)	8.79 ms	4
java.io.File.**exists** ()		0.485 ms (0%)	0.509 ms	10
java.util.Scanner.**<clinit>**		0.467 ms (0%)	6.0 ms	1
java.util.regex.Pattern.**group0** ()		0.445 ms (0%)	2.68 ms	54
java.io.WinNTFileSystem.**canonicalize** (String)		0.440 ms (0%)	0.688 ms	4
java.util.jar.JarFile.**initializeVerifier** ()		0.433 ms (0%)	0.864 ms	1
sun.nio.cs.UTF_8$Encoder.**encodeArrayLoop** (java.nio.CharBuffer, java.nio.ByteBuffer)		0.396 ms (0%)	0.501 ms	107
java.text.DecimalFormat.**<init>** (String, java.text.DecimalFormatSymbols)		0.359 ms (0%)	0.915 ms	1
java.lang.ProcessEnvironment.**<clinit>**		0.358 ms (0%)	2.5 ms	1
java.util.regex.Pattern.**createGroup** (boolean)		0.357 ms (0%)	0.392 ms	35
java.util.regex.Pattern.**expr** (java.util.regex.Pattern.Node)		0.349 ms (0%)	7.12 ms	39
java.util.HashMap.**putVal** (int, Object, Object, boolean, boolean)		0.332 ms (0%)	0.660 ms	406

Complejidad de implementación HashSet

Realizando varias corridas con diferentes cargas de ingresos se determinó que la complejidad del HashSet es **O (1)** es decir constante, ya que el rango de tiempo estuvo ente 54.647 y 57.171 pero esta implementación no superó esos valores (ejecutándolo varias veces en el mismo rango de carga). Este resultado concuerda con la teoría ya que este ofrece de manera continua el rendimiento del tiempo aunque no garantiza el orden de los elementos.