

# **Randomize Your Tests**

**...and it will blow your socks off**

**Dawid WEISS**  
Carrot Search s.c.

# **Introduction**



## **Motivation**

**(why are unit tests insufficient?)**



## **Randomized Testing: Core Concepts**



## **RT in Practice (Example)**



## **Summary and Conclusions**

# Introduction

# Historical note

## Industry and academia

Duran, Ntafos: An evaluation of random testing, 1984! Haskell: QuickCheck.

## Hacking.

Fuzzifiers, vulnerability discovery.

## Lucene Test Framework.

Lots of great ideas. Driven by real needs and bugs. Many contributors.

## Carrot Search.

Handcrafted pseudo-randomness in Carrot2, Lingo3G, HPPC, ...

**Motivation**

# Software Tests

**Confidence in implementation.**

**Reliability.**

**Insurance against future changes (regressions).**

**Assumptions  $\neq$  The Real World**

# Can this ever fail?

```
String [] words = {"Berlin", "Spring", "Sun" };  
Random r = new Random();  
while (true) {  
    String buzzword = words[Math.abs(r.nextInt()) % words.length];  
    System.out.println(buzzword);  
}
```



# Can this ever fail?

```
String [] words = {"Berlin", "Spring", "Sun" };
Random r = new Random();
while (true) {
    String buzzword = words[Math.abs(r.nextInt()) % words.length];
    System.out.println(buzzword);
}
```

ArrayIndexOutOfBoundsException

$\text{Math.abs}(\text{Integer.MIN\_VALUE}) < 0$  (!)

**“Anything that can go wrong will go wrong.”**

**(An unlikely event will eventually occur.)**

# Randomized Testing

**Randomized testing is **not** a paradigm shift!**

(Does not replace common-sense unit tests.)

**Randomized testing is **not** mutation-based!**

(To increase coverage, like PiTest.)

# Randomized Testing

(mini-manifesto)

**1. Each run covers a different execution path.**

By substituting components or diversifying data.

**2. Each run is deterministic.**

Can be repeated from the same initial randomization seed.

**3. Tests are repeated lots of times.**

Not just on changes, continuous build server, developers.

Why not just...  
**test everything?**

# What can be randomized?

## 1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

```
// Lucene/Solr source code (LTC)
final int iters = atLeast(200);
for (int iter = 0; iter < iters; iter++) {
    // ...
}
```

```
// Lucene/Solr source code (LTC)
if (random.nextBoolean()) {
    term = _TestUtil.randomRealisticUnicodeString(random);
} else {
    // we want to mix in limited-alphabet symbols so
    // we get more sharing of the nodes given how few
    // terms we are testing...
    term = simpleRandomString(random);
}
```



# What can be randomized?

## 1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

## 2. Software components.

If multiple implementations exist. LTC: Field, Directory, IndexSearcher...

```
public static String randomDirectory(Random random) {
    if (rarely(random)) {
        return CORE_DIRECTORIES[random.nextInt(CORE_DIRECTORIES.length)];
    } else {
        return "RAMDirectory";
    }
}
```

# What can be randomized?

## 1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

## 2. Software components.

If multiple implementations exist. LTC: Field, Directory, IndexSearcher...

## 3. Environment.

Locale, TimeZone, JVM (!), operating system.

```
locale = TEST_LOCALE.equals("random") ? randomLocale(random) : ...
Locale.setDefault(locale);
timeZone = TEST_TIMEZONE.equals("random") ? randomTimeZone(random) : ...
TimeZone.setDefault(timeZone);
```

# What can be randomized?

## 1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

## 2. Software components.

If multiple implementations exist. LTC: Field, Directory, IndexSearcher...

## 3. Environment.

Locale, TimeZone, JVM (!), operating system.

## 4. Exceptional triggers.

I/O problems, network problems (using mocks or runtime engineering).

# Assertions in randomized test code?

## **Compare against reference.**

Naïve, previous or alternative implementations.

## **Sanity checks.**

Crude output checks (boundary conditions). Sanity assertions inside code.

## **Nothing.**

Unchecked exceptions. Or a jvm core dump. Surprisingly effective :)

# Randomized Testing in Practice

# Lucene Q/A Components

## **Uwe's forbidden API checker.**

<https://code.google.com/p/forbidden-apis/>

## **smoketester.py**

Pre-release automated sanity checks.

## **Runtime checks (rules) in LTC.**

Static memory leaks, stray threads, temporary file cleanups, assertions...

## **Security manager.**

For enforcing filesystem (and other) policies.

## **RandomizedRunner.**

<https://github.com/carrotsearch/randomizedtesting/>

## RandomizedRunner's goals

# Compatibility

with JUnit (and tools). At 99%, relax contracts when useful.

# Built-in randomization

including reporting/ stack augmentations.

## Test isolation

by tracking spawned threads. Timeouts. Terminations.

## Utilities

@Repeat, @Seed, @Nightly, @TestGroup, @TestFactories...

**EXAMPLE**



```
@RunWith(RandomizedRunner.class) // <--- USE RR!  
public class TestClass1 {  
    @Test public void test1_1() { }  
    @Test public void test1_2() { }  
    @Test public void test1_3() { }  
}  
  
public class TestClass2 extends TestClass1 {  
    @Test public void test2_1() { }  
    @Test public void test2_2() { }  
    @Test public void test2_3() { }  
}
```

com.carrotsearch.randomizedtesting.examples.TestClass2 [Runner: JUnit 4]

- test2\_3 {seed=[68508C8145637F4:6F5AC393F4DC201B]} (0.013 s)
- test1\_2 {seed=[68508C8145637F4:21C04FFE26CC71AA]} (0.008 s)
- test2\_1 {seed=[68508C8145637F4:4F3143B3C6BCCCC21]} (0.004 s)
- test1\_1 {seed=[68508C8145637F4:C8D97BD78E4607BF]} (0.003 s)
- test2\_2 {seed=[68508C8145637F4:CF0918A9EF5A933F]} (0.003 s)
- test1\_3 {seed=[68508C8145637F4:44418D9217A50763]} (0.004 s)

com.carrotsearch.randomizedtesting.examples.TestClass2 [Runner: JUnit 4]

- test1\_2 {seed=[98233F305B18EE47:BF6678066982A819]} (0.013 s)
- test1\_1 {seed=[98233F305B18EE47:567F4C2FC108DE0C]} (0.007 s)
- test1\_3 {seed=[98233F305B18EE47:DAE7BA6A58EBDED0]} (0.005 s)
- test2\_3 {seed=[98233F305B18EE47:F1FCF46BBB92F9A8]} (0.004 s)
- test2\_2 {seed=[98233F305B18EE47:51AF2F51A0144A8C]} (0.003 s)
- test2\_1 {seed=[98233F305B18EE47:D197744B89F21592]} (0.005 s)

com.carrotsearch.randomizedtesting.examples.TestClass2 [Runner: JUnit 4]

- test2\_3 {seed=[90E59ACD5471DAC4:F93A5196B4FBCD2B]} (0.012 s)
- test1\_3 {seed=[90E59ACD5471DAC4:D2211F975782EA53]} (0.008 s)
- test1\_2 {seed=[90E59ACD5471DAC4:B7A0DDFB66EB9C9A]} (0.005 s)
- test2\_2 {seed=[90E59ACD5471DAC4:59698AACAF7D7E0F]} (0.004 s)
- test2\_1 {seed=[90E59ACD5471DAC4:D951D1B6869B2111]} (0.003 s)
- test1\_1 {seed=[90E59ACD5471DAC4:5EB9E9D2CE61EA8F]} (0.006 s)

```
@RunWith(RandomizedRunner.class)
public class TestMe {
    @Test
    public void theTruthIsOutThere() {
        RandomizedContext ctx = RandomizedContext.current();
        Random rnd = ctx.getRandom(); // => no setSeed()!
        // Your pseudo-randomized code here.
        assertTrue(rnd.nextBoolean());
    }
}
```

```
@RunWith(RandomizedRunner.class)
public class TestMe {
    @Test
    public void theTruthIsOutThere() {
        RandomizedContext ctx = RandomizedContext.current();
        Random rnd = ctx.getRandom(); // => no setSeed()!
        // Your pseudo-randomized code here.
        assertTrue(rnd.nextBoolean());
    }
}
```

```
public class TestMe extends RandomizedTest {
    @Test
    public void theTruthIsOutThere() {
        // Your pseudo-randomized code here.
        assertTrue(randomBoolean());
    }
}
```

```
@RunWith(RandomizedRunner.class)
public class TestMe {
    @Test
    public void theTruthIsOutThere() {
        RandomizedContext ctx = RandomizedContext.current();
        Random rnd = ctx.getRandom(); // => no setSeed()!
        // Your pseudo-randomized code here.
        assertTrue(rnd.nextBoolean());
    }
}
```


```
public class TestMe extends RandomizedTest {
    @Test
    public void theTruthIsOutThere() {
        // Your pseudo-randomized code here.
        assertTrue(randomBoolean());
    }
}
```


- **The Random is assigned per-thread (increases reproducibility).**
- **The context is invalidated after each scope (suite, test).**
- **Randomness is derived from the master seed, method name, repetition.**

Runs: 1/1

✖ Errors: 0

✖ Failures: 0

▶  TestMe [Runner: JUnit 4] (0.001 s)

 theTruthIsOutThere (0.001 s)

Runs: 1/1

Errors: 0

Failures: 1

TestMe [Runner: JUnit 4] (0.001 s)

theTruthIsOutThere (0.002 s)

Failure Trace



```
java.lang.AssertionError
  at __randomizedtesting.SeedInfo.seed([A83B9F3F1764DE62:AF30FDD461E80521]:0)
  at org.junit.Assert.fail(Assert.java:92)
  at org.junit.Assert.assertTrue(Assert.java:43)
  at org.junit.Assert.assertTrue(Assert.java:54)
```

**Synthetic stack trace frame with the master:method seed.**

**In case the JVM freezes, use a signal or jstack to get the master seed.**

```
> jstack 6502
```

```
...  
"TEST-TestScope-TestHanging.myTest-seed#[96F549A92265CBF8]"  
waiting on condition [1169ef000]  
  java.lang.Thread.State: TIMED_WAITING (sleeping)  
at java.lang.Thread.sleep(Native Method)  
at TestHanging.myTest(TestHanging.java:26)
```



**Once you've hit a failure...**

```
@Seed("A83B9F3F1764DE62")
public class TestMe extends RandomizedTest {
    @Test
    @Repeat(iterations = 20)
    public void theTruthIsOutThere() {
        // Your pseudo-randomized code here.
        assertTrue(randomBoolean());
    }
}
```

## ...or set the seed externally via properties...

```
-Dtests.seed=A83B9F3F1764DE62
-Dtests.iter=20
```

TestMe [Runner: JUnit 4] (0.032 s)

theTruthIsOutThere (0.032 s)

- theTruthIsOutThere {#0 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#1 seed=[A83B9F3F1764DE62:1B664128552ACE0D]} (0.
- theTruthIsOutThere {#2 seed=[A83B9F3F1764DE62:958FD7F404EE86C6]} (0.
- theTruthIsOutThere {#3 seed=[A83B9F3F1764DE62:A4617C116810DDEF]} (0
- theTruthIsOutThere {#4 seed=[A83B9F3F1764DE62:E8A0F9BCC9181D54]} (0
- theTruthIsOutThere {#5 seed=[A83B9F3F1764DE62:795A2AE3B4A45054]} (0.
- theTruthIsOutThere {#6 seed=[A83B9F3F1764DE62:47844E65A6944052]} (0.C
- theTruthIsOutThere {#7 seed=[A83B9F3F1764DE62:DB37D41F8580D4FC]} (0
- theTruthIsOutThere {#8 seed=[A83B9F3F1764DE62:E99B3171F24BC3A6]} (0.
- theTruthIsOutThere {#9 seed=[A83B9F3F1764DE62:3E1067CB961CF4F4]} (0.
- theTruthIsOutThere {#10 seed=[A83B9F3F1764DE62:CB518F9044A0D62C]} (
- theTruthIsOutThere {#11 seed=[A83B9F3F1764DE62:40F64355C03D77E5]} (0
- theTruthIsOutThere {#12 seed=[A83B9F3F1764DE62:27C5D6EC2540B514]} ((
- theTruthIsOutThere {#13 seed=[A83B9F3F1764DE62:488EF1F3B9D53464]} (0
- theTruthIsOutThere {#14 seed=[A83B9F3F1764DE62:1510FE6B6BA4723D]} ((
- theTruthIsOutThere {#15 seed=[A83B9F3F1764DE62:76A2136F7926E70C]} (0
- theTruthIsOutThere {#16 seed=[A83B9F3F1764DE62:F059B4A6B22E8C65]} ((
- theTruthIsOutThere {#17 seed=[A83B9F3F1764DE62:441666BD7E1BFE17]} (0
- theTruthIsOutThere {#18 seed=
- theTruthIsOutThere {#19 seed=

20 repetitions from the same master seed

TestMe [Runner: JUnit 4] (0.029 s)

theTruthIsOutThere (0.029 s)

- theTruthIsOutThere {#0 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#1 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#2 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#3 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#4 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#5 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#6 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#7 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#8 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#9 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
- theTruthIsOutThere {#10 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#11 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#12 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#13 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#14 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#15 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#16 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#17 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#18 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#19 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((
- theTruthIsOutThere {#20 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} ((

20 repetitions from the same master/ method seed

# The downsides of test randomization

## **Failure reproducibility.**

Hard to achieve for concurrent, time-based or otherwise uncoordinated code.

## **Incompatible random component configurations.**

May require ignoring certain tests at runtime.

## **Randomized tests are typically pretty hairy.**

More difficult to debug. Can cause JVM indigestion...

```
public void testCuriousMassiveString() throws Exception {
    String text = "yqt \u0728\u0707\u0712\u0720\u0734 \u204c\u201d hyipy \u2667\u2619" +
        "\u26ec\u267b\u26da uboyjwfbv \u2012\u205d\u2042\u200a\u2047\u2040 gyxmmz yvv %" +
        "\ufb86 \n<script> hupvobbv jvsztd x rww ct{1,5} brteyi dfgyzqbm hdykd ahgeizyhv" +
        " kLn c#\ud8f8\udd74 fPezd ktedq \ufcea=\ud997\uddc9\u876bJ\u0231\u098a\uddce\u0f872" +
        " zquqah \ub9d6\u144e\uc686 3\u093d\udfca\u1215\u0614 tnorask \u0348\u0334\u0334" +
        "\u0300\u033d geqdeoghh foojebut \ufb52\u227ag\u09bd\u0c3a\u07efk nyantr lksxw fs" +
        "zies ubzqfolksjjpgk \u6fa3\u0859\u0c36\u0501\ucca0\u0306\u001e\ua756\u052f \ucaf7" +
        "\u0247\u0009\u0fdddg \ud83c\udd02\u083c\uddaf \u05628\u2b49\u02e3\u0718\u0769\u4f1b" +
        "\u0496\u0766\uuecaa\u0fb44 \u001d \u0006hr\u00f0\u0e649\u041a\u0da6f\u0dfa5\u0f31b\u0e274" +
        " ptgjf \ud8cc\u0df83M\u0013\u04c6i \u205f\u2004\u2032\u2001\u2057\u2066 \u07d0\u0acdb" +
        "\u06a5z pqfxwgbwe \ud1bc\u2eba\u2d45\u02ee\u56df xnujtfS \u1b19\u1b17\u1b39\u1b20" +
        "\u1b69\u1b58\u1b03\u1b6e\u1b73\u1b20 afsL xLzZiqh ahrhckhktf \ud801\u0c5b\u0801\u0c61" +
        " bkpmeyyqobwi qnkunmpjpihezll plhhws \u37f7\u41d6\u3dca\u3e80\u4923\u36b8 \u195a\u1959" +
        "\u196f\u1968\u1958\u197f </p \u0006s\u019f\u0da82\u0c90H_\u079d\u0fd6f: idpp \u217c" +
        "\u2168\u2185\u2179\u2156\u2159\u217c\u2154\u2158 ({1,5})( jkieylqzmb bfirmaj \uea71" +
        "\uf17f\u0749\u054c \ud9ea\u0df83\u0d8ea\u0e91j x\u3366\u09c2\u0828\u0d13~@\u06fda\u0e00" +
        " \ud834\u0c03\u0834\u0c2b\u0834\u0c8b\u0834\u0c08 d1 \ud802\u0e3a\u0802\u0e36\u0802" +
        "\u0e23\u0802\u0e56 \u20ad\u20a0\u20a0\u20a0\u20b8\u20b4\u20ad lcql \u0547\u0156]" +
        "\u0e344V] \ud90c\u0d7d\u000e5\u0f965\u0f15e\u008f qdn \udac3\u0e3c buy m qni \u31a4\u31a3" +
        "\u31b2\u31a9\u31a7 \u20df\u20ee\u20d3\u20d4\u20f1\u20de \u4dc1\u4dc4\u4dde\u4dec\u4df9" +
        "\u4dee\u4dc5 \u0db4\u0c36 gStfqnfWY \u24ab\u08d69\u0301 n?jv||ji )- \u0db0b\u0e77\u0f634" +
        "\u0762 tPKkjLbcntsk eebtzirw xo hktxy \n</ vxro xpr mtlp p|tjfi?.- lxpfo \u0dbd7" +
        "\u0df78\u0dbf5\u0df57 u..b jjj]p?e jtckvhqb \u20eb\u20ea\u20fa\u20ef\u20e1\u20ed\u20eb vvm" +
        "uhbsvyi jx mqwxckf \u00d7 qqzs \u05ae\u022d\u0db7c\u0dfb1\u070b vnhkz egnutyuat \u0daa2" +
        "\u0df20\u0fa45#0\u2b61\u0d0e \u09a2\u0996\u09cd\u09b4 v \u0dbdb\u0e9bqp owsln \u0a837\u0a833" +
        "\u0a83f\u0a83f\u0a83f\u0a831\u0a83c\u0a839 \u01a15\u01a1c\u01a12 \ud83c\u0e20 >&#x>129 \ud9f1" +
        "\u0df8c\u0eccd \ud809\u0c48\u0809\u0c72 wswskop \u0a70c
```

## hotspot escape analysis bug in FreqProxTermsWriterPerField::flush

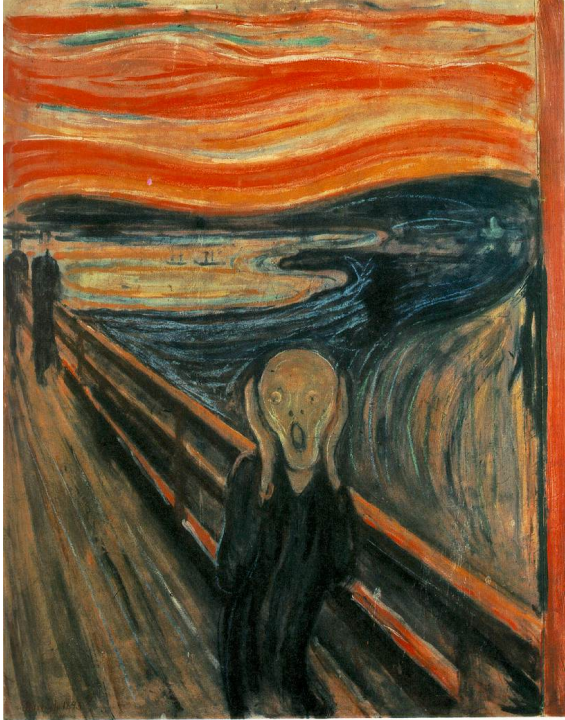
```
org.apache.lucene.index.FreqProxTermsWriterPerField::flush (1207 bytes)
  @ 4 org.apache.lucene.index.FieldInfo::isIndexed (5 bytes) accessor
  @ 16 org.apache.lucene.codecs.perfield.PerFieldPostingsFormat$FieldsWriter::addField (327 bytes) too big
  @ 23 org.apache.lucene.codecs.BlockTreeTermsWriter$TermsWriter::getComparator (4 bytes) inline (hot)
    @ 0 org.apache.lucene.util.BytesRef::getUTF8SortedAsUnicodeComparator (4 bytes) inline (hot)
    @ 32 org.apache.lucene.index.FieldInfo::getIndexOptions (5 bytes) accessor
    @ 61 java.lang.Enum::compareTo (44 bytes) too big
    @ 79 java.lang.Enum::compareTo (44 bytes) too big
    @ 97 java.lang.Enum::compareTo (44 bytes) too big
    @ 238 java.util.HashMap::size (5 bytes) accessor
    @ 267 org.apache.lucene.index.TermsHashPerField::sortPostings (9 bytes) executed < MinInlin
    @ 279 org.apache.lucene.util.BytesRefHash::size (5 bytes) accessor
    @ 288 org.apache.lucene.util.BytesRef::<init> (8 bytes) inline (hot)
      @ 4 org.apache.lucene.util.BytesRef::<init> (9 bytes) inline (hot)
        @ 5 org.apache.lucene.util.BytesRef::<init> (41 bytes) inline (hot)
          @ 1 java.lang.Object::<init> (1 bytes) inline (hot)
            @ 26 org.apache.lucene.util.BytesRef::isValid (329 bytes) hot method too big
        @ 309 org.apache.lucene.index.ByteSliceReader::<init> (5 bytes) inline (hot)
          @ 1 org.apache.lucene.store.DataInput::<init> (5 bytes) inline (hot)
            @ 1 java.lang.Object::<init> (1 bytes) inline (hot)
          @ 318 org.apache.lucene.index.ByteSliceReader::<init> (5 bytes) inline (hot)
            @ 1 org.apache.lucene.store.DataInput::<init> (5 bytes) inline (hot)
              @ 1 java.lang.Object::<init> (1 bytes) inline (hot)
            @ 331 org.apache.lucene.index.SegmentInfo::getDocCount (23 bytes) inline (hot)
            @ 334 org.apache.lucene.util.FixedBitSet::<init> (29 bytes) inline (hot)
              @ 1 org.apache.lucene.search.DocIdSet::<init> (5 bytes) inline (hot)
                @ 1 java.lang.Object::<init> (1 bytes) inline (hot)
                  @ 11 org.apache.lucene.util.FixedBitSet::bits2words (17 bytes) inline (hot)
              @ 350 org.apache.lucene.index.Term::<init> (13 bytes) inline (hot)
                @ 6 org.apache.lucene.util.BytesRef::<init> (8 bytes) inline (hot)
```

```
@ 1 java.lang.Object::<init> (1 bytes) inline (hot)
@ 26 org.apache.lucene.util.BytesRef::isValid (329 bytes) hot method too big
@ 9 org.apache.lucene.index.Term::<init> (15 bytes) inline (hot)
@ 1 java.lang.Object::<init> (1 bytes) inline (hot)
@ 393 org.apache.lucene.util.ByteBlockPool::setBytesRef (107 bytes) inline (hot)
@ 405 org.apache.lucene.index.TermsHashPerField::initReader (87 bytes) inline (hot)
@ 83 org.apache.lucene.index.ByteSliceReader::init (153 bytes) inline (hot)
@ 427 org.apache.lucene.index.TermsHashPerField::initReader (87 bytes) inline (hot)
@ 83 org.apache.lucene.index.ByteSliceReader::init (153 bytes) inline (hot)
@ 434 org.apache.lucene.codecs.BlockTreeTermsWriter$TermsWriter::startTerm (18 bytes)
inline (hot)
@ 500 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 1154 org.apache.lucene.codecs.TermStats::<init> (15 bytes) inline (hot)
@ 1 java.lang.Object::<init> (1 bytes) inline (hot)
@ 1157 org.apache.lucene.codecs.BlockTreeTermsWriter$TermsWriter::finishTerm (87 bytes)
inline (hot)
@ 30 org.apache.lucene.util.fst.Util::toIntsRef (55 bytes) already compiled into a medium
@ 37 org.apache.lucene.util.fst.NoOutputs::getNoOutput (4 bytes) inline (hot)
@ 40 org.apache.lucene.util.fst.Builder::add (767 bytes) too big
@ 52 org.apache.lucene.util.BytesRef::deepCopyOf (15 bytes) inline (hot)
@ 4 org.apache.lucene.util.BytesRef::<init> (8 bytes) inline (hot)
@ 4 org.apache.lucene.util.BytesRef::<init> (9 bytes) inline (hot)
@ 5 org.apache.lucene.util.BytesRef::<init> (41 bytes) inline (hot)
@ 1 java.lang.Object::<init> (1 bytes) inline (hot)
@ 26 org.apache.lucene.util.BytesRef::isValid (329 bytes) hot method too big
@ 10 org.apache.lucene.util.BytesRef::copyBytes (64 bytes) inline (hot)
@ 52 java.lang.System::arraycopy (0 bytes) (intrinsic)
@ 56 org.apache.lucene.codecs.BlockTreeTermsWriter$PendingTerm::<init> (16 bytes)
inline (hot)
@ 2 org.apache.lucene.codecs.BlockTreeTermsWriter$PendingEntry::<init> (10 bytes)
inline (hot)
@ 1 java.lang.Object::<init> (1 bytes) inline (hot)
@ 563 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
@ 1 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
```



```
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 610 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
@ 1 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 657 org.apache.lucene.index.SegmentInfo::getDocCount (23 bytes) inline (hot)
@ 710 org.apache.lucene.util.FixedBitSet::set (84 bytes) inline (hot)
@ 763 org.apache.lucene.index.SegmentInfo::getDocCount (23 bytes) call site not reached
@ 848 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
@ 1 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
```

```
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 871 org.apache.lucene.store.DataInput::readVInt (114 bytes) too big
@ 888 org.apache.lucene.util.BytesRef::<init> (8 bytes) call site not reached
@ 927 org.apache.lucene.util.BytesRef::grow (34 bytes) call site not reached
@ 942 org.apache.lucene.index.ByteSliceReader::readBytes (84 bytes) too big
@ 975 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
@ 1 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
@ 7 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
@ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) NodeCountInliningCutoff
@ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) NodeCountInliningCutoff
@ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) NodeCountInliningCutoff
@ 985 org.apache.lucene.store.DataInput::readVInt (114 bytes) size > DesiredMethodLimit
@ 1077 org.apache.lucene.codecs.lucene40.Lucene40PostingsWriter::addPosition (385 bytes)
size > DesiredMethodLimit
@ 1114 org.apache.lucene.codecs.lucene41.Lucene41PostingsWriter::addPosition (367 bytes)
size > DesiredMethodLimit
@ 1200 org.apache.lucene.util.FixedBitSet::cardinality (15 bytes) size > DesiredMethodLimit
@ 1203 org.apache.lucene.codecs.BlockTreeTermsWriter.TermsWriter::finish (354 bytes)
size > DesiredMethodLimit
```



Software Failure. Press left mouse button to continue.

Guru Meditation #00000025.65045338

# **Summary and Conclusions**

# The Good

**Explore complex boundary conditions.**

May or may not hit them, but there is a chance!

**Unexpected component-component interactions.**

Pairwise component compatibility.

**Unexpected environment interactions.**

JVM, operating system differences.

**Production-grade tool support.**

RandomizedTesting used in Lucene, Solr, ES...

# The Bad

**Failures may prove difficult to debug.**

Race conditions. Complex inputs. Multithreaded code.

# The Ugly

**Looping with different master seed.**

Currently requires hacks.

**Tight testing sandbox.**

WTF moments may be frustrating.





dawid.weiss@carrotsearch.com

**Randomized testing package is @labs:  
<http://labs.carrotsearch.com/randomizedtesting.html>**