



JavaOneSM

Sun's 2000 Worldwide Java Developer Conference*



JavaOneSM
Sun's 2000 Worldwide Java Developer Conference™

High-Performance Computing With Java™ Technology

Stefan Nilsson

Assistant Professor

Royal Institute of Technology

Rules for Optimization (Trad.)

- Don't
- Use the right algorithms and data structures
- 90/10-rule
- Use a profiler
- Measure before and after
- (Don't optimize as you go)



Optimization

- Algorithms and data structures
- Code tuning



Code Lives Longer Than Hardware

- TeX
- Y2K
- Genetic programs



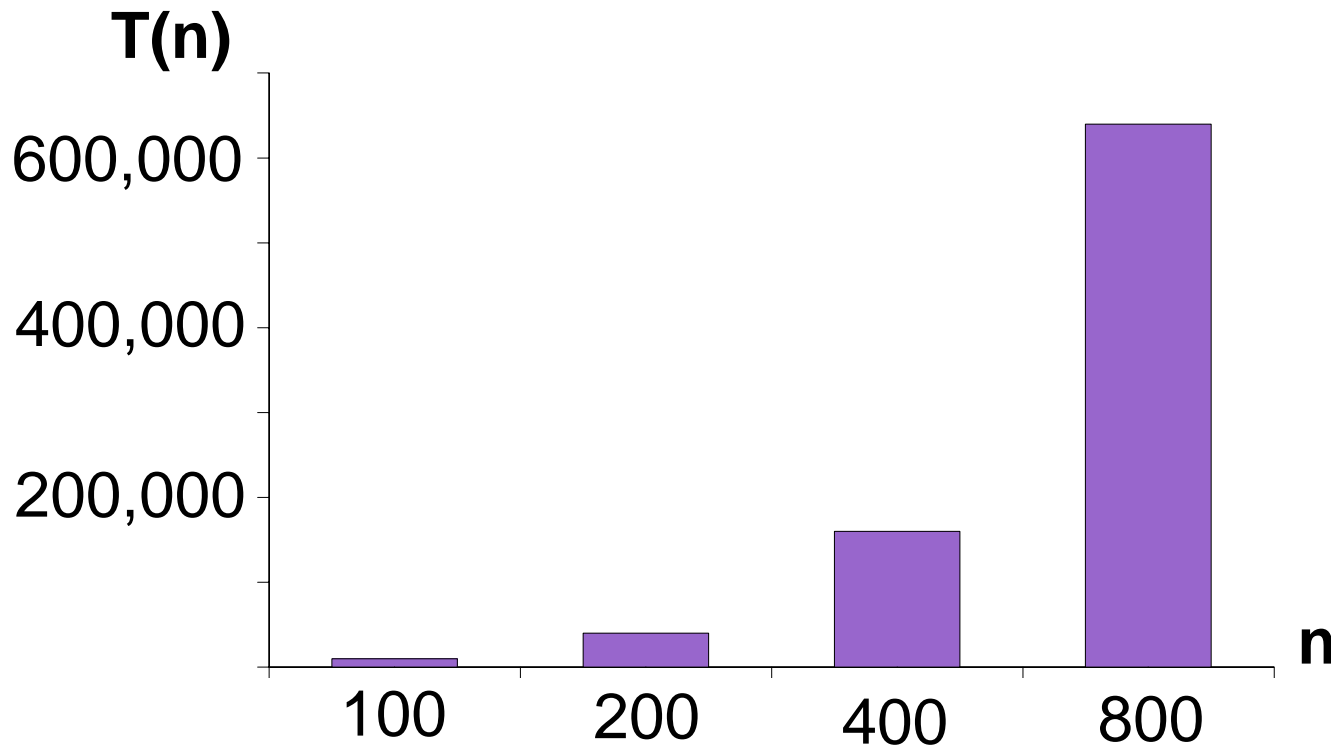
A Changing World

- Processors get faster
- Memory gets larger
- Problems grow larger
- Programs **either** improve or degrade



Program Degradation Due to Quadratic Growth

$T(n) = n^2$, where n is size and $T(n)$ time



Spot the Square

```
String str = "";  
  
for (int i = 0; i < 10000; i++) {  
    str += i;  
}  
  
System.out.println(str);
```



Remove the Square

```
StringBuffer strBuf = new StringBuffer();  
  
for (int i = 0; i < 10000; i++) {  
    strBuf.append(i);  
}  
  
System.out.println(strBuf);
```



What Really Happened?

```
> javap -c MyClass
```

```
...
```

```
 8 new #3 <Class StringBuffer>
11 dup
12 invokespecial #4 <Method StringBuffer()>
15 aload_1
16 invokevirtual #5 <Method StringBuffer append(String)>
19 iload_2
20 invokevirtual #6 <Method StringBuffer append(int)>
23 invokevirtual #7 <Method String toString()>
26 astore_1
27 iinc 2 1
30 iload_2
31 sipush 10000
34 if_icmplt 8
```

```
...
```



Spot the Square (2)

```
List myList = new ArrayList();  
Random rand = new Random();  
  
for (int i = 1; i < 75000; i++) {  
    int k = rand.nextInt(i);  
    myList.add(k, new Object());  
}
```



Remove the Square (2)

```
List myList = new IndexTreeList();  
Random rand = new Random();  
  
for (int i = 1; i < 75000; i++) {  
    int k = rand.nextInt(i);  
    myList.add(k, new Object());  
}
```

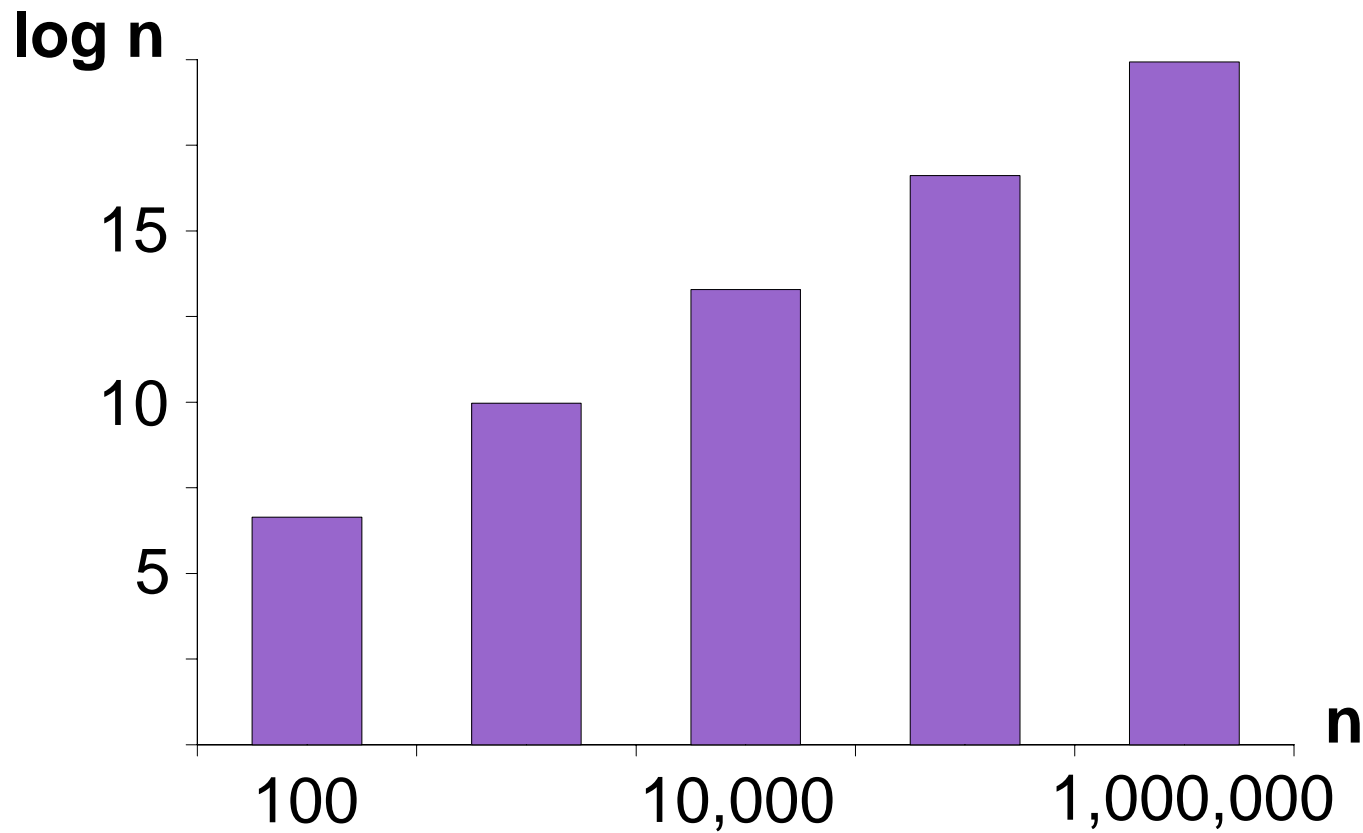


List Operations

	Array	Linked	IndexTree
add(i, x)	n	n	log n
remove(i)	n	n	log n
get(i)	1	n	log n
get(0)	1	1	log n

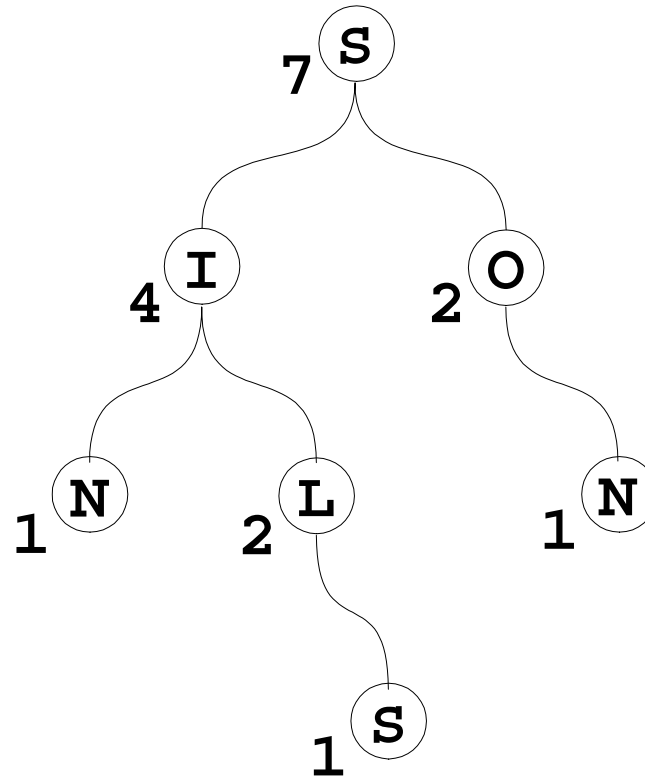


Logarithmic Growth



Index Tree

0 N
1 I
2 L
3 S
4 S
5 O
6 N



Object Pool

```
class Node {
    private static Node[] pool = new Node[50];
    private static int topOfPool = 0;
    ...

    public static Node create() {
        if (topOfPool > 0)
            return pool[--topOfPool];
        else
            return new Node();
    }

    public static void destroy(Node n) { }
}
```



Spot the Square (3)

[Trick Question]

```
StringBuffer strBuf = new StringBuffer();
BigInteger k;

for (int i = 0; i < 25000; i++)
    strBuf.append("7");
k = new BigInteger(strBuf.toString());
k = k.multiply(k);
System.out.println(k);
```



Use a Profiler

```
> java -Xrunhprof ClassName
```

```
TRACE 259:
```

```
java.math.MutableBigInteger.divide(<Unknown>:Unknown line)  
java.math.BigInteger.toString(<Unknown>:Unknown line)  
java.math.BigInteger.toString(<Unknown>:Unknown line)  
java.lang.String.valueOf(<Unknown>:Unknown line)
```



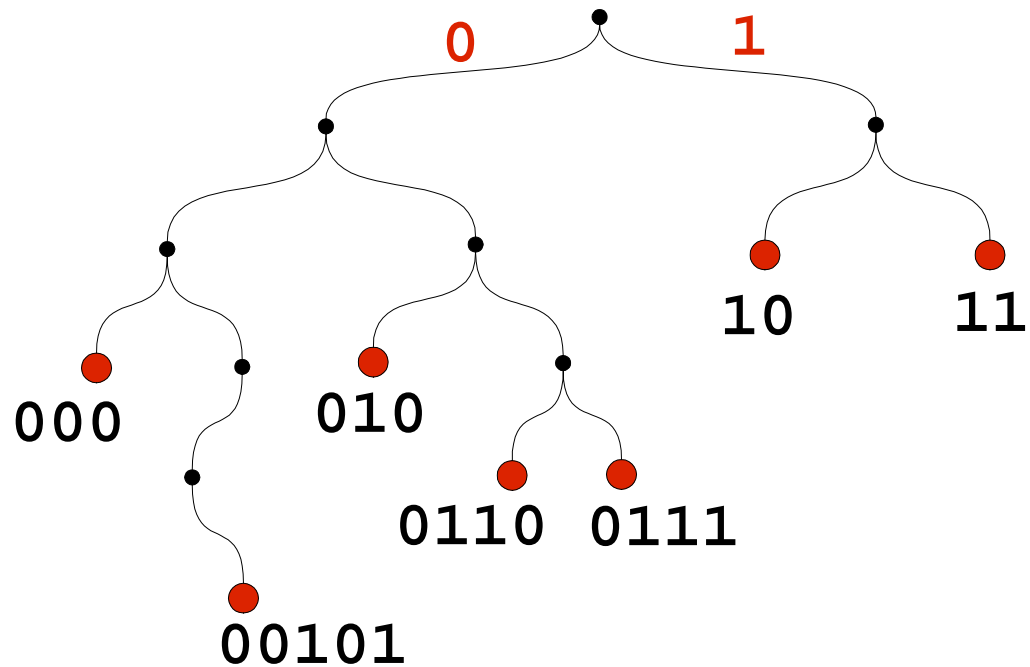
Alt: Perform Explicit Timings

```
PrintStream o = System.out;  
  
o.print("multiply: ");  
long time = System.currentTimeMillis();  
k = k.multiply(k);  
o.println(System.currentTimeMillis()-time);  
  
o.print("toString: ");  
time = System.currentTimeMillis();  
String str = k.toString();  
o.println(System.currentTimeMillis()-time);
```



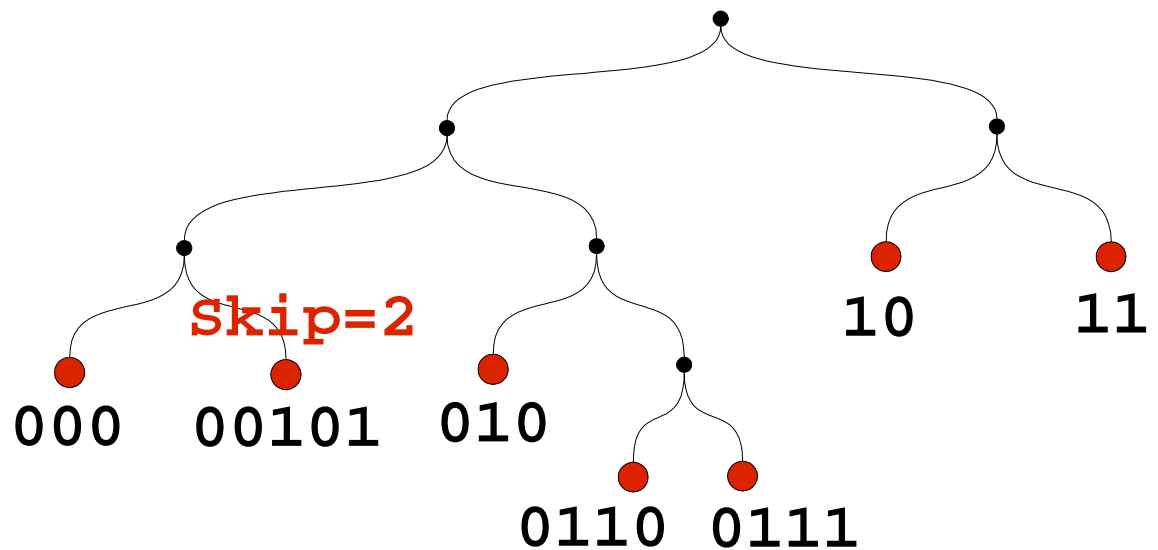
Routing Table as Trie

000
00101
010
0110
0111
10
11



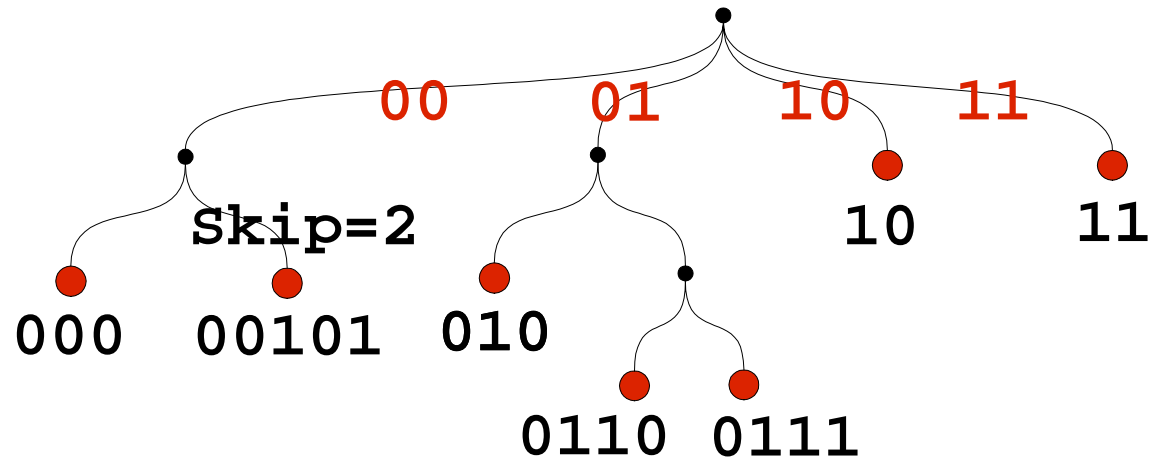
Path Compression

000
00101
010
0110
0111
10
11



Level Compression

000
00101
010
0110
0111
10
11



Routing Table Lookup

```
int node = trie[0];
int pos = getSkip(node);
int branch = getBranch(node);
int adr = getAdr(node);
while (branch != 0) {
    node = trie[adr + extract(pos, branch, s)];
    pos += branch + getSkip(node);
    branch = getBranch(node);
    adr = getAdr(node);
}
```



References

- <http://www.research.ibm.com/journal/sj39-1.html>
- <http://www.javagrande.org>
- <http://www.nada.kth.se/~snilsson>





JavaOneSM

Sun's 2000 Worldwide Java Developer Conference*