

# Table Of Content

---

|   |    |
|---|----|
| <a href="#">ca.ubc.cs.kisynski.bell</a> ..... | 2  |
| <a href="#">Bell</a> .....                    | 2  |
| <a href="#">_Test</a> .....                   | 9  |
| <a href="#">Index</a> .....                   | 12 |

# Package ca.ubc.cs.kisynski.bell

## Class Summary

### [Bell](#)

This package provides functions which are useful while dealing with set partitions.

### [Test](#)

Simple demo of Bell package.

---

ca.ubc.cs.kisynski.bell

## Class Bell

```
java.lang.Object
|
+--ca.ubc.cs.kisynski.bell.Bell
```

---

< [Methods](#) >

---

```
public class Bell
extends java.lang.Object
```

This package provides functions which are useful while dealing with set partitions. We provide (hopefully) fast methods for sets of size up to 15 and methods with no set size restrictions which use BigInteger objects. The later ones are constrained only by the available memory size and the expected life length of the user.

The package is named after Eric Temple Bell (1883 - 1960), see Bell, E. T. "Exponential Numbers." Amer. Math. Monthly 41, 411-419, 1934.

This file is part of ca.ubc.cs.kisynski.bell package.

ca.ubc.cs.kisynski.bell package is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

ca.ubc.cs.kisynski.bell package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with ca.ubc.cs.kisynski.bell package; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

### Author:

Jacek Kisynski

### Version:

1.2 03-April-2006

\*\*\*\*\*

- Changes by Jacek Kisynski:
- use of generics (update to J2SE 5.0)

## Methods

### bigGrowthString

```
public static int[] bigGrowthString(java.math.BigInteger i,  
                                     int n)  
    throws java.lang.IllegalArgumentException
```

Returns i-th set partition of the set of n elements encoded as restricted growth strings.

**Parameters:**

i - number of the requested partition (starting from 0)  
n - number of elements in the set.

**Returns:**

restricted growth strings representing partition.

**Throws:**

java.lang.IllegalArgumentException - if n is less than zero or i is not a correct partition number.

---

### bigIndexAfterRemove

```
public static java.math.BigInteger bigIndexAfterRemove(java.math.BigInteger  
index,  
                                                       int length,  
                                                       int toRemove)  
    throws java.lang.IllegalArgumentException
```

Computes the new index of the growth string after removing the particular element.

**Parameters:**

index - of the growth string  
length - before removal  
toRemove - element (counting from 0)

**Returns:**

new index Method calls bigGrowthString(), normalizeString() and bigIndexOf(). They all could be combine into one, but I don't think that cost savings are worth doing this.

**Throws:**

java.lang.IllegalArgumentException - if length is less than two or index of an element to remove is outside of a proper range or if growth string's index is not correct..

---

## bigIndexAndDataAfterRemove

```
public static java.math.BigInteger[]  
bigIndexAndDataAfterRemove( java.math.BigInteger index,  
                                int length,  
                                int toRemove)  
    throws java.lang.IllegalArgumentException
```

Computes the new index of the growth string after removing the particular element together with information about blocks.

### Parameters:

index - of the growth string  
length - before removal  
toRemove - element (counting from 0)

### Returns:

new index and if element wasn't the only one in the block -1, else number of other blocks  
Method calls `bigGrowthString()`, `normalizeString()` and `bigIndexOf()`. They all could be combine into one, but I don't think that cost savings are worth doing this.

### Throws:

`java.lang.IllegalArgumentException` - if length is less than two or index of an element to remove is outside of a proper range or if growth string's index is not correct.

---

## bigIndexOf

```
public static java.math.BigInteger bigIndexOf(int[] string)  
    throws java.lang.IllegalArgumentException
```

Returns index of the set partition (given as a restricted growth string).

### Parameters:

string - representing partition

### Returns:

index of the partition (starting from 0)

### Throws:

`java.lang.IllegalArgumentException` - if string is not a proper growth string.

---

## bigIndexesAfterInsert

```
public static java.math.BigInteger[]  
bigIndexesAfterInsert(java.math.BigInteger index,  
                        int length,  
                        int insertionIndex)  
    throws java.lang.IllegalArgumentException
```

Computes an array of new indexes of the growth string after inserting a new element at position insertionIndex. If resulting array has length n, then first (n - 1) growth strings have (n-2) blocks and the n-th one has (n - 1) blocks.

### Parameters:

index - of the growth string  
length - before insertion  
insertionIndex - of the new element (counting from 0)

### Returns:

new indexes Method calls bigGrowthString(), normalizeString() and bigIndexOf(). They all could be combine into one, but I don't think that cost savings are worth doing this.

### Throws:

java.lang.IllegalArgumentException - if insertion index is not within the right scop or length is less than zero or index is not consistent with the length.

---

## bigNumber

```
public static java.math.BigInteger bigNumber(int n)  
    throws java.lang.IllegalArgumentException
```

Computes n-th Bell number using Bell triangle. To compute n-th Bell number for the first time we use two arrays of size n + 1 and n and perform  $(n^2 - n) / 2$  additions. Once we have computed n-th Bell number, all predeccessing Bell numbers are computed and stored in memory to avoid unnecessary recomputation.

### Parameters:

n - , works even if n = 1000, for n = 10000 I ran out of memory ;-)

### Returns:

n-th Bell number.

### Throws:

java.lang.IllegalArgumentException - if n is less than zero.

---

## growthString

```
public static int[] growthString(int i,  
                                  int n)  
    throws java.lang.IllegalArgumentException
```

Returns i-th set partition of the set of n elements encoded as restricted growth strings. Method for practical use.

**Parameters:**

i - number of the requested partition (starting from 0)  
n - number of elements in the set.

**Returns:**

restricted growth strings representing partition.

**Throws:**

java.lang.IllegalArgumentException - if n is less than zero or i is not a correct partition number.

---

## growthStrings

```
public static int[][] growthStrings(int n)  
    throws java.lang.IllegalArgumentException
```

Returns set partitions of the set of n elements encoded as restricted growth strings. Method for practical use. Based on algorithm by George Hutchinson presented in CACM 6 (1963), 613-614 and described in Donald E. Knuth, TAOC Vol 7, pre-fascicle 3B, 25-27. For complexity analysis see exercise 46 in Knuth's book.

**Parameters:**

n - number of elements in the set.

**Returns:**

restricted growth strings representing set partitions.

**Throws:**

java.lang.IllegalArgumentException - if n is less than zero or greater than fifteen.

---

## growthStringsIterator

```
public static java.util.Iterator growthStringsIterator(int n)
```

Iterator over restricted growth strings.

**Parameters:**

n - number of elements in the set.

**Returns:**

an iterator over the partitions of the set represented as restricted growth strings. This code is based on the Iterator code in java.util.AbstractList.

---

## indexAfterRemove

```
public static int indexAfterRemove(int index,  
                                   int length,  
                                   int toRemove)  
    throws java.lang.IllegalArgumentException
```

Computes the new index of the growth string after removing the particular element. Method for practical use.

**Parameters:**

index - of the growth string  
length - before removal  
toRemove - element (counting from 0)

**Returns:**

new index

**Throws:**

java.lang.IllegalArgumentException - if length is less than two or index of an element to remove is outside of a proper range or if growth string's index is not correct.

---

## indexAndDataAfterRemove

```
public static int[] indexAndDataAfterRemove(int index,  
                                              int length,  
                                              int toRemove)  
    throws java.lang.IllegalArgumentException
```

Computes the new index of the growth string after removing the particular element together with information about partitions. Method for practical use.

**Parameters:**

index - of the growth string  
length - before removal  
toRemove - element (counting from 0)

**Returns:**

new index and if element wasn't the only one in the block -1, else number of other blocks.

**Throws:**

java.lang.IllegalArgumentException - if length is less than two or index of an element to remove is outside of a proper range or if growth string's index is not correct.

---

## indexOf

```
public static int indexOf(int[] string)
    throws java.lang.IllegalArgumentException
```

Returns index of the set partition (given as a restricted growth string). Method for practical use.

**Parameters:**

string - representing partition

**Returns:**

index of the partition (starting from 0)

**Throws:**

java.lang.IllegalArgumentException - if string is too long or it is not a proper growth string.

---

## indexesAfterInsert

```
public static int[] indexesAfterInsert(int index,
    int length,
    int insertionIndex)
    throws java.lang.IllegalArgumentException
```

Computes an array of new indexes of the growth string after inserting a new element at position insertionIndex. If resulting array has length n, then first (n - 1) growth strings have (n - 2) blocks and the n-th one has (n - 1) blocks. Method for practical use.

**Parameters:**

index - of the growth string

length - before insertion

insertionIndex - of the new element (counting from 0)

**Returns:**

new indexes

**Throws:**

java.lang.IllegalArgumentException - if insertion index is not within the right scope or length is less than zero or index is not consistent with the length.

---

## initialize

```
public static void initialize()
    throws java.io.IOException
```

Reads precomputed data into arrays.

**Throws:**

java.io.IOException - if resource "ca.ubc.cs.kisynski.Bell.data" is not available.

---

## normalizeString

```
public static void normalizeString(int[] string)
    throws java.lang.IllegalArgumentException
```

Normalizes restricted growth string after removal or insertion of some elements. Method works in linear time with respect to the original string length. It uses additional array of the size  $\max(\text{string}[]) + 1$ .

**Parameters:**

string - to be normalized.

**Throws:**

java.lang.IllegalArgumentException - if string contains negative numbers.

---

## number

```
public static int number(int n)
    throws java.lang.IllegalArgumentException
```

N-th Bell number. Method for practical use.

**Parameters:**

n -

**Returns:**

n-th Bell number.

**Throws:**

java.lang.IllegalArgumentException - if n is outside of the desired range.

---

ca.ubc.cs.kisynski.bell

## Class \_Test

```
java.lang.Object
|
+-- java.awt.Component
    |
    +-- java.awt.Container
        |
        +-- java.awt.Panel
            |
            +-- java.applet.Applet
                |
                +-- javax.swing.JApplet
                    |
                    +-- ca.ubc.cs.kisynski.bell._Test
```

**All Implemented Interfaces:**

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer

---

< [Constructors](#) > < [Methods](#) >

---

```
public class _Test
extends javax.swing.JApplet
```

Simple demo of Bell package.

This file is part of ca.ubc.cs.kisynski.bell package.

ca.ubc.cs.kisynski.bell package is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

ca.ubc.cs.kisynski.bell package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with ca.ubc.cs.kisynski.bell package; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

**Author:**

Jacek Kisynski

**Version:**

1.0 24-April-2005

## Constructors

### **\_Test**

```
public _Test()
```

## Methods

### **init**

```
public void init()
```

This method initializes this applet.

**Overrides:**

init in class java.applet.Applet

---

# main

```
public static void main(java.lang.String[] args)
```

Main method so it can be run as an application.

**Parameters:**

args - not used

# INDEX

## B

[bigGrowthString](#) ... 3  
[bigIndexAfterRemove](#) ... 3  
[bigIndexAndDataAfterRemove](#) ... 4  
[bigIndexesAfterInsert](#) ... 5  
[bigIndexOf](#) ... 4  
[bigNumber](#) ... 5  
[Bell](#) ... 2

## G

[growthString](#) ... 6  
[growthStrings](#) ... 6  
[growthStringsIterator](#) ... 6

## I

[indexAfterRemove](#) ... 7  
[indexAndDataAfterRemove](#) ... 7  
[indexesAfterInsert](#) ... 8  
[indexOf](#) ... 8  
[init](#) ... 10  
[initialize](#) ... 8

## M

[main](#) ... 11

## N

[normalizeString](#) ... 9  
[number](#) ... 9