1.3 BAGS, QUEUES, AND STACKS

▸ stacks

▸ resizing arrays

▸ queues

▸ generics

▸ *iterators*

▸ applications

Robert Sedgewick | Kevin Wayne

http://algs4.cs.princeton.edu

Iteration

Design challenge. Support iteration over stack items by client, without revealing the internal representation of the stack.

s[]	it	was	the	best	of	times	*null*	*null*	*null*	*null*
	0	1	2	3	4	5	6	7	8	9

i   N

first   current

times → of → best → the → was → it → *null*

Java solution. Make stack implement the `java.lang.Iterable` interface.

43

Iterators

Q. What is an `Iterable` ?

A. Has a method that returns an `Iterator`.

Q. What is an `Iterator` ?

A. Has methods `hasNext()` and `next()`.

Q. Why make data structures `Iterable` ?

A. Java supports elegant client code.

```
public interface Iterable<Item>
{
    Iterator<Item> iterator();
}
```

```
public interface Iterator<Item>
{
    boolean hasNext();
    Item next();
    void remove();
}
```
optional; use
at your own risk

"foreach" statement (shorthand)

```
for (String s : stack)
    StdOut.println(s);
```

equivalent code (longhand)

```
Iterator<String> i = stack.iterator();
while (i.hasNext())
{
    String s = i.next();
    StdOut.println(s);
}
```

Stack iterator: linked-list implementation

```java
import java.util.Iterator;

public class Stack<Item> implements Iterable<Item>
{
    ...

    public Iterator<Item> iterator() { return new ListIterator(); }

    private class ListIterator implements Iterator<Item>
    {
        private Node current = first;

        public boolean hasNext() {  return current != null;  }
        public void remove()      {  /* not supported */      }
        public Item next()
        {
            Item item = current.item;
            current    = current.next;
            return item;
        }
    }
}
```
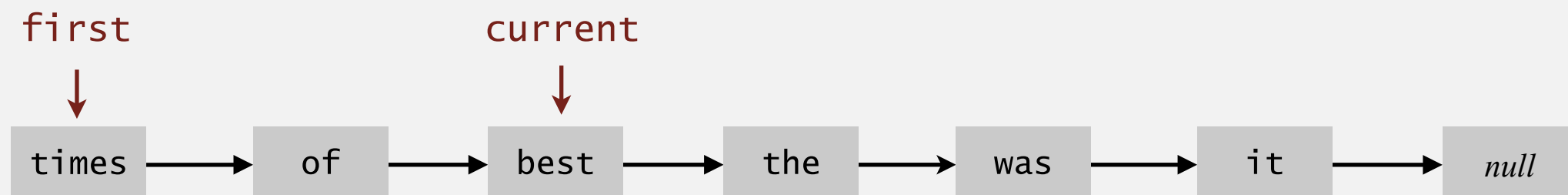
throw UnsupportedOperationException

throw NoSuchElementException
if no more items in iteration

first

current

times		of		best		the		was		it		*null*

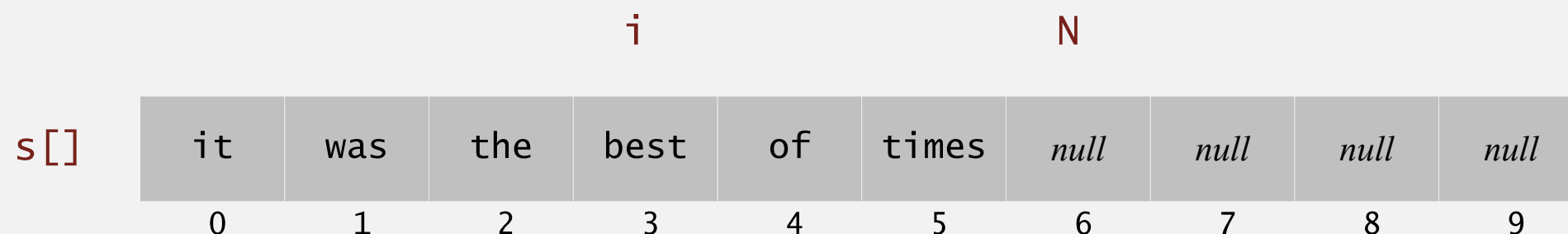# Stack iterator: array implementation

```java
import java.util.Iterator;

public class Stack<Item> implements Iterable<Item>
{
    …

    public Iterator<Item> iterator()
    { return new ReverseArrayIterator(); }

    private class ReverseArrayIterator implements Iterator<Item>
    {
        private int i = N;

        public boolean hasNext() {  return i > 0;         }
        public void remove()      {  /* not supported */  }
        public Item next()        {  return s[--i];        }
    }
}
```

i   N

s[]	it	was	the	best	of	times	null	null	null	null
	0	1	2	3	4	5	6	7	8	9

Iteration: concurrent modification

Q. What if client modifies the data structure while iterating?

A. A fail-fast iterator throws a `java.util.ConcurrentModificationException`.

concurrent modification

```
for (String s : stack)
    stack.push(s);
```

Q. How to detect?

A.

- Count total number of `push()` and `pop()` operations in `Stack`.
- Save counts in `*Iterator` subclass upon creation.
- If, when calling `next()` and `hasNext()`, the current counts do not equal the saved counts, throw exception.