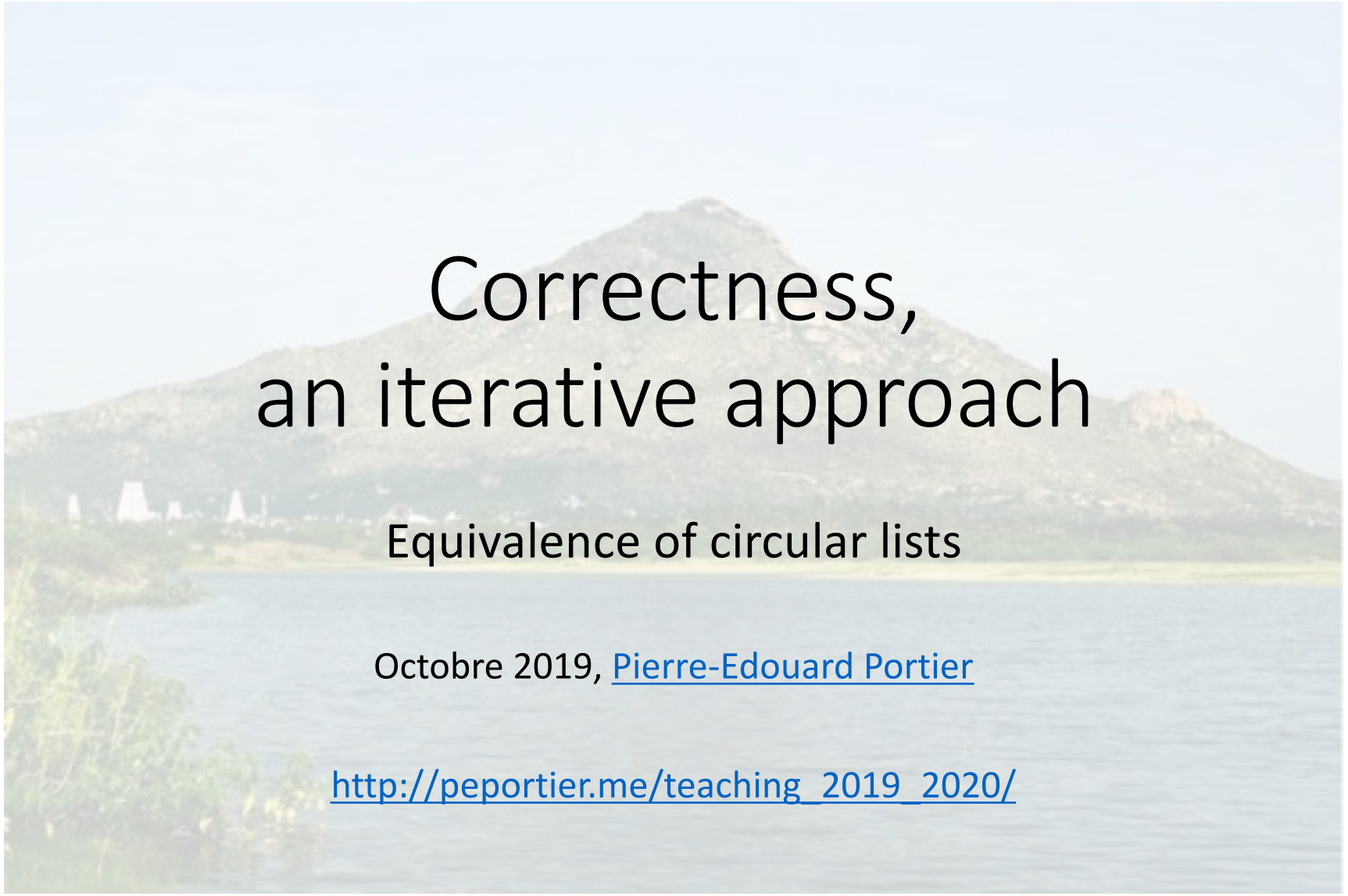


Correctness, an iterative approach

Equivalence of circular lists

Octobre 2019, Pierre-Edouard Portier

http://peportier.me/teaching_2019_2020/

$$A(i: 0 \leq i < N)$$
$$B(i: 0 \leq i < N)$$
$$A.i = A.(i + N)$$

```
public class TestCL {
    static CL<Integer> cl_a, cl_b, cl_big_a, cl_big_b;

    @BeforeClass
    public static void setup() {
        cl_a = new CL<Integer>();
        Collections.addAll(cl_a, 7, 8, 9);

        cl_b = new CL<Integer>();
        Collections.addAll(cl_b, 8, 9, 7);

        int size = 1000000;
        cl_big_a = new CL<Integer>(size);
        cl_big_b = new CL<Integer>(size);
        for(int i = 0; i < size ; i++) {
            cl_big_a.add(i); cl_big_b.add(i);
        }
        cl_big_b.set_rot(size-1);
    }
```

cl_a = (7,8,9)

```
@Test
public void test_set_get() {
    assertEquals((Integer)8, cl_a.get(7));
    cl_a.set(4, 6);
    assertEquals((Integer)6, cl_a.get(1));
    cl_a.set(1, 8);
}
```

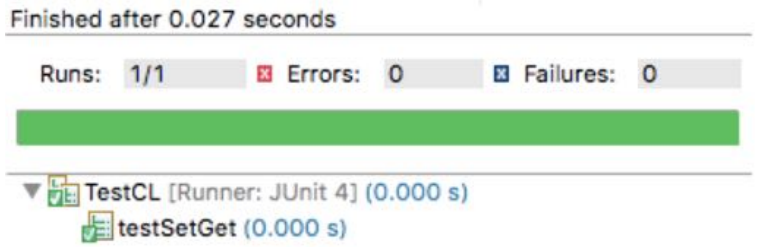
```java
public class CL<E> extends ArrayList<E> {

    private static final long serialVersionUID = 23698120493447273356L;

    private int circular_index(int index) {
        return (size() != 0) ? index % size() : index;
    }

    @Override
    public E get(int index) {
        return super.get(circular_index(index));
    }

    @Override
    public E set(int index, E element) {
        return super.set(circular_index(index), element);
    }
```


Finished after 0.027 seconds
Runs: 1/1
Errors: 0
Failures: 0
TestCL [Runner: JUnit 4] (0.000 s)
testSetGet (0.000 s)

Set of rotations of A:

$$RA.i = A(k : i \leq k < i + N)$$
$$RA.i = RA.(i + N)$$
$$\#RA \leq N$$

cl_a = (7,8,9)

```
@Test
public void test_rot() {
    cl_a.set_rot(1);
    assertEquals((Integer)8, cl_a.get(0));
    assertEquals((Integer)9, cl_a.get(1));
    assertEquals((Integer)7, cl_a.get(2));
    cl_a.set_rot(0);
}
```

```
private int rot = 0;

private int circular_index(int index) {
    return (size() != 0) ? (index + rot) % size() : index;
}
```

```
public void set_rot(int i) {
    rot = i;
}
```

Postcondition:

$$R: \quad res \equiv \ (\exists i, j :: RA.i = RB.j)$$

cl_a = (7,8,9)

cl_b = (8,9,7)

```
@Test
public void test_rot_eq() {
    cl_a.set_rot(1);
    assertTrue(cl_a.get_eq(cl_b));
    cl_a.set_rot(0);
}
```

```java
public boolean get_eq(CL<E> other) {
    if(this.size() != other.size()) return false;
    boolean ok = true;
    for(int i = 0 ; i < this.size() ; i++) {
        ok = ok && this.get(i).equals(other.get(i));
        if(!ok) break;
    }
    return ok;
}
```

$$R: \quad res \equiv \ (\exists i, j :: RA.i = RB.j)$$

Naïve solution:
Compare A with each element of RB

```java
@Test
public void test_naive_eq() {
    assertTrue(cl_a.naive_eq(cl_b));
}
```

```
public boolean naive_eq(CL<E> other) {
    if(this.size() != other.size()) return false;
    int old_rot = rot;
    for(int i = 0 ; i < this.size() ; i++) {
        this.set_rot(i);
        if(this.get_eq(other)) {
            this.set_rot(old_rot);
            return true;
        }
    }
    this.set_rot(old_rot);
    return false;
}
```

Time complexity of the naïve solution is:
$O(N^2)$

```
@Test(timeout=1000)
public void test_naive_eq_scalability() {
    assertTrue(cl_big_a.naive_eq(cl_big_b));
}
```

RA and RB are either disjoint or equal

$$\big(\forall\, i, j, k ::\;\; RA.i = RB.j \quad \Rightarrow \quad RA.(i+k) = RB.(j+k)\big)$$

It is sufficient to compare canonical elements

AA: first element of RA by lexicographical ordering

R can be solved by computing AA and BB

Lexicographically minimal string rotation

Naïve algorithm in $O(N^2)$

To find the solution $res \equiv true$, we can:
(a) Find a pair (i, j) such that $RA.i = RB.j$

To find the solution $res \equiv false$, we can:
(b) Observe $AA \neq BB$

To find the solution $res \equiv true$, we can:
(a) Find a pair (i, j) such that $RA.i = RB.j$

Weakening of (a) into a loop invariant:

$$P\colon\ 0 \le h \quad \wedge \quad (\forall\, k : 0 \le k < h : RA.i.k = RB.j.k)$$

$$P \wedge h \ge N \quad \Rightarrow \quad true \equiv\ (\exists\, i, j :: RA.i = RB.j)$$

To find the solution $res \equiv false$, we can:
(b) Observe $AA \neq BB$

Weakening (b) into a loop invariant:

$$QA\colon\ 0 \leq i \quad \wedge \quad (\forall\, k : 0 \leq k < i : RA.k > BB)$$

$$QA \wedge i \geq N \quad \Rightarrow \quad false \equiv\ (\exists\, i, j :: RA.i = RB.j)$$

Symmetrically for QB
The two CL differ if:

$$QA \wedge QB \wedge (i \geq N\ \vee\ j \geq N)$$

First sketch of a solution…

```
public boolean eq(CL<E> other) {
    boolean res = false;
    int h,i,j;
    h = i = j = 0;
    while(h < size() && i < size() && j < size()) {
        // increase h+i+j while maintaining P && QA && QB
    }
    if(h >= size()) res = true;
    else if(i >= size() || j >= size()) res = false;
    return res;
}
```

$$P\colon\ 0 \le h \quad \wedge \quad (\forall\, k : 0 \le k < h : RA.i.k = RB.j.k)$$

When $RA.i.h = RB.j.h$ (i.e., $A.(i+h) = B.(j+h)$):

$h \leftarrow h + 1$ will:

(a) maintain P, and

(b) assure progress by increasing $h + i + j$
if i and j are themselves not modified

$$P\colon\ 0 \le h \quad \wedge \quad (\forall\, k : 0 \le k < h : RA.i.k = RB.j.k)$$
$$QA\colon\ 0 \le i \quad \wedge \quad (\forall\, k : 0 \le k < i : RA.k > BB)$$

A: (7,7,9,6) B: (7,7,6,9)

h=2, i=0, j=0

$$RA.i.h > RB.j.h$$
$\Longrightarrow$ {P, déf. Ordre lexicographique}
$$RA.i > RB.j$$
$\Longrightarrow$ {déf. BB}
$$RA.i > BB$$

Therefore, $i \leftarrow i + 1$ will maintain QA.
Then, $h \leftarrow 0$ is necessary to maintain P.
However, for $h + i + j$ to increase,
we must do at least $i \leftarrow i + h + 1$.

$$P\colon\ 0 \le h \quad \wedge \quad (\forall\, k : 0 \le k < h : RA.i.k = RB.j.k)$$

$$QA\colon\ 0 \le i \quad \wedge \quad (\forall\, k : 0 \le k < i : RA.k > BB)$$

$$RA.i.h > RB.j.h \quad \wedge \quad P$$

⇒ {def. P, $0 \le p \le h$}

$$RA.i.h > RB.j.h \quad \wedge \quad (\forall\, k : p \le k < h : RA.i.k = RB.j.k)$$

⇒ {lexicographic ordering}

$$RA.(i+p) > RB.(j+p)$$

⇒ {def. BB}

$$RA.(i+p) > BB$$

$$QA \quad \wedge \quad P \quad \wedge \quad RA.i.h > RB.j.h$$

⇒ {see above}

$$QA \quad \wedge \quad (\forall\, p : 0 \le p \le h : RA.(i+p) > BB)$$

= {renaming the bounded variable: $i + p = k$}

$$QA \quad \wedge \quad (\forall\, k : i \le k \le i + h : RA.k > BB)$$

= {def. QA}

$$QA[i \backslash \mathrm{i} + h + 1]$$

A: (7,7,9,6) B: (7,7,6,9)

h=2, i=0, j=0

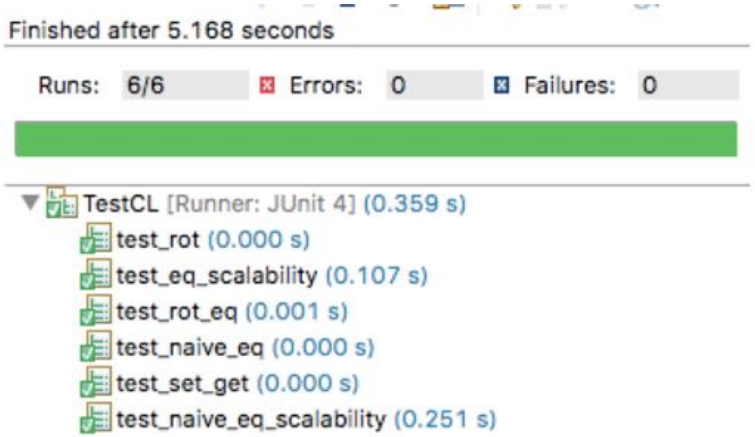
h=0, i=3, j=0

```
public class CL<E extends Comparable<E>> extends ArrayList<E> {

public boolean eq(CL<E> other) {
    int h,i,j;
    h = i = j = 0;
    while(h < size() && i < size() && j < size()) {
        int compare_res = this.get(i+h).compareTo(other.get(j+h));
        if( compare_res == 0) {
            h = h+1;
        }
        else if(compare_res > 0) {
            i = i+h+1; h = 0;
        }
        else if(compare_res < 0) {
            j = j+h+1; h = 0;
        }
    }
    return h >= size();
}
```

```
@Test(timeout=1000)
public void test_naive_eq_scalability() {
    assertTrue(cl_big_a.naive_eq(cl_big_b));
}

@Test(timeout=1000)
public void test_eq_scalability() {
    assertTrue(cl_big_a.eq(cl_big_b));
}
```


Finished after 5.168 seconds
Runs: 6/6
Errors: 0
Failures: 0
TestCL [Runner: JUnit 4] (0.359 s)
test_rot (0.000 s)
test_eq_scalability (0.107 s)
test_rot_eq (0.001 s)
test_naive_eq (0.000 s)
test_set_get (0.000 s)
test_naive_eq_scalability (0.251 s)

Verify test coverage.

Shiloach, Yossi.
"A fast equivalence-checking algorithm for circular lists."
Information Processing Letters 8.5 (1979): 236-238.

Gasteren, Antonetta JM.
"On the shape of mathematical arguments. "
Vol. 445. *Springer Science & Business Media*, 1990.

https://en.wikipedia.org/wiki/Lexicographically_minimal_string_rotation