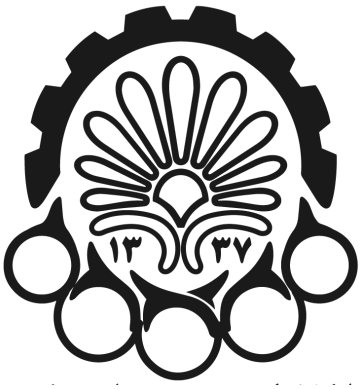


# تمرین چهارم درس آزمون نرم افزار


**دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)**

بردیا اردکانیان

۹۸۳۱۰۷۲


استاد درس: دکتر گوهری

بهار ۱۴۰۲

7.2.2

Q3.

Subsumption has a significant weakness. Suppose criterion $C_{strong}$ subsumes criterion $C_{weak}$ and that test set $T_{strong}$ satisfies $C_{strong}$ and test set $T_{weak}$ satisfies $C_{weak}$. It is not necessarily the case that $T_{weak}$ is a subset of $T_{strong}$. It is also not necessarily the case that $T_{strong}$ reveals a fault if $T_{weak}$ reveals a fault. Explain these facts.

در اینجا مشکل این است که استدلال بر عملکرد قوی‌تر Cstrong نسبت به Cweak به تنهایی نشانگر رابطه زیرمجموعه بین Tweak و Tstrong نیست. به طور کلی، در انتخاب تست برای برآورده کردن یک نیاز تست خاص، چندین گزینه وجود دارد. با فرض وجود یک نیاز تست مشترک برای هر دو Cweak وCstrong ، امکان انتخاب یک تست برای Tweak و یک تست دیگر برای Tstrong وجود دارد. به طور خاص، می‌توان یک تست که خطا را آشکار می‌کند را برای Tweak انتخاب کرد، اما آن را برای Tstrong انتخاب نکرد.

Q5.

Answer questions a–g for the graph defined by the following sets: $N$ = {1, 2, 3, 4, 5, 6, 7}

$N_0$ = {1}

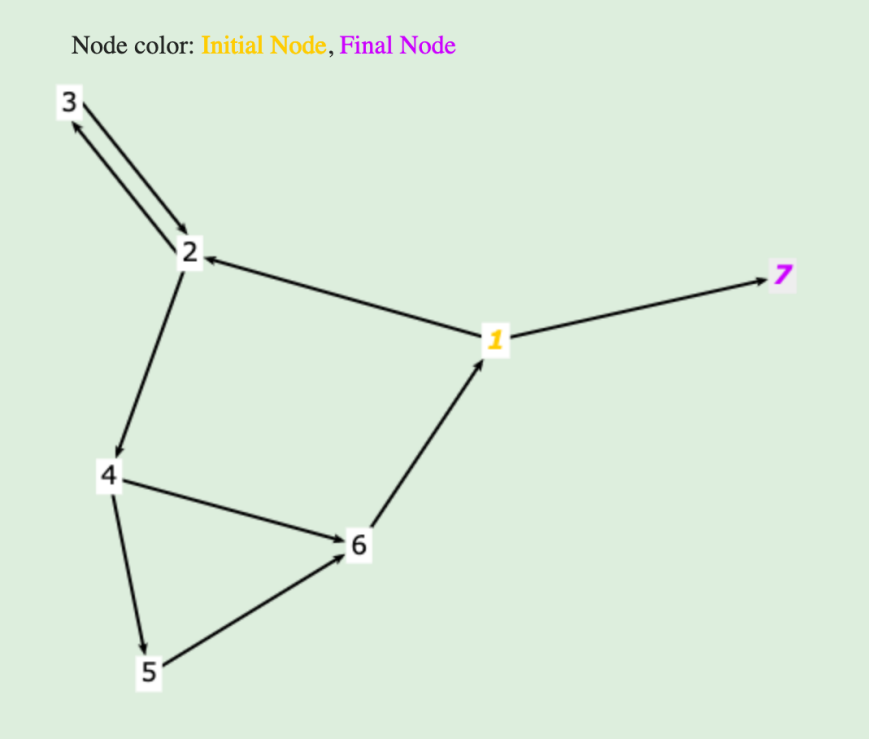
$N_f$= {7}

$E$ = {(1, 2), (1, 7), (2, 3), (2, 4), (3, 2), (4, 5), (4, 6), (5, 6), (6, 1)} Also consider the following (candidate) test paths:

$p_1$ =[1,2,4,5,6,1,7]
$p_2$ =[1,2,3,2,4,6,1,7] $p_3$ =[1,2,3,2,4,5,6,1,7]

(a) Draw the graph.

با کمک ابزار موجود در سایت کتاب رسم می‌کنیم.




(b) List the test requirements for Edge-Pair Coverage. (Hint: You should get 12 requirements of length 2.)

{[1, 2, 3], [1, 2, 4], [2, 3, 2], [2, 4, 5], [2, 4, 6], [3, 2, 3], [3, 2, 4], [4, 5, 6], [4, 6, 1], [5, 6, 1], [6, 1, 2], [6, 1, 7]}

(c) Does the given set of test paths satisfy Edge-Pair Coverage? If not, state what is missing.

هیچکدام {[3, 2, 3], [6, 1, 2]} را cover نمی‌کنند.

(d) Consider the simple path **[3, 2, 4, 5, 6]** and test path **[1, 2, 3, 2, 4, 6, 1, 2, 4, 5, 6, 1, 7]**. Does the test path tour the simple path directly? With a sidetrip? If so, write down the sidetrip.

با $[4, 6, 1, 2, 4]$ می‌شود.

(e) List the test requirements for Node Coverage, Edge Coverage, and Prime Path Coverage on the graph.

$NC$: $\{1, 2, 3, 4, 5, 6, 7\}$

$EC$: $\{(1,2), (1,7), (2,3), (2,4), (3,2), (4,5), (4,6), (6,1), (5,6)\}$

$PPC$: {[1, 2, 4, 5, 6, 1], [1, 2, 4, 6, 1], [2, 4, 6, 1, 2], [2, 4, 5, 6, 1, 2], [3, 2, 4, 6, 1, 7], [3, 2, 4, 5, 6, 1, 7], [4, 6, 1, 2, 4], [4, 5, 6, 1, 2, 4], [4, 6, 1, 2, 3], [4, 5, 6, 1, 2, 3], [5, 6, 1, 2, 4, 5], [6, 1, 2, 4, 6], [6, 1, 2, 4, 5, 6], [3, 2, 3], [2, 3, 2]}

(f) List test paths from the given set that achieve Node Coverage but not Edge Coverage on the graph.

[1, 2, 3, 2, 4, 5, 6, 1, 7]

(g) List test paths from the given set that achieve Edge Coverage but not Prime Path Coverage on the graph.

[1, 2, 3, 2, 4, 5, 6, 1, 7], [1, 2, 4, 6, 1, 7]

7.2.3

Q1.

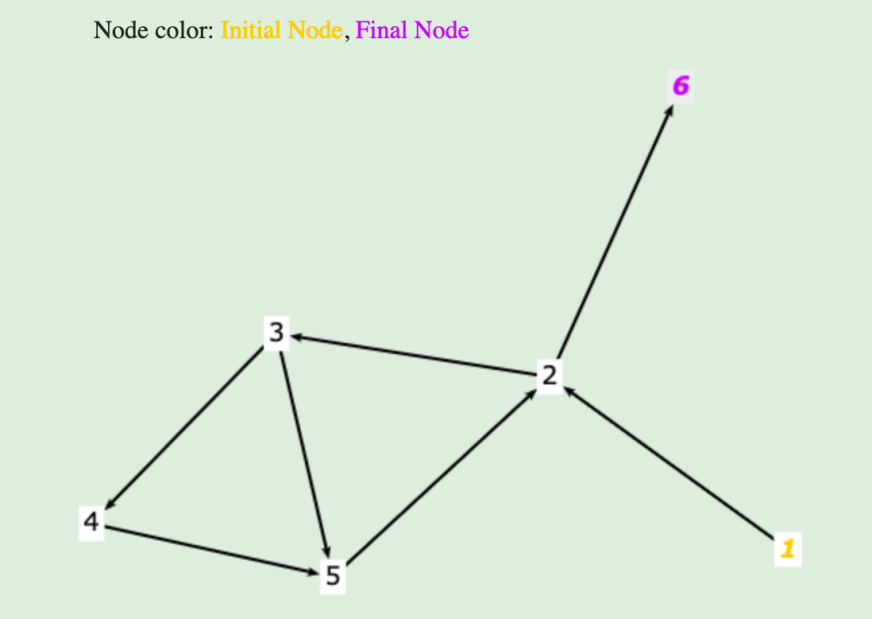
Below are four graphs, each of which is defined by the sets of nodes, initial nodes, final nodes, edges, and defs and uses. Each graph also contains some test paths. Answer the following questions about each graph.

گراف ۲

**Graph II.**

$N = \{1, 2, 3, 4, 5, 6\}$
$N_0 = \{1\}$
$N_f = \{6\}$
$E = \{(1,2), (2,3), (2,6), (3,4), (3,5), (4,5), (5,2)\}$
$def(1) = def(3) = use(3) = use(6) = \{x\}$
// Assume the use of x in 3 precedes the def
**Test Paths:**
$t1 = [1, 2, 6]$
$t2 = [1, 2, 3, 4, 5, 2, 3, 5, 2, 6]$
$t3 = [1, 2, 3, 5, 2, 3, 4, 5, 2, 6]$
$t4 = [1, 2, 3, 5, 2, 6]$

a)



b)

۶ مسیر du برای x وجود دارد.

1. [1, 2, 3] 2. [1, 2, 6] 3. [3, 4, 5, 2, 3] 4. [3, 4, 5, 2, 6] 5. [3, 5, 2, 3] 6. [3, 5, 2, 6]

c)

| | **direct** | **sidetrip** |
|---|---|---|
| **T1** | 2 | No |
| **T2** | 1, 3, 6 | No |
| **T3** | 1, 4, 5 | No |
| **T4** | 1, 6 | No |

d)

$$\{t_2, t_3, t_4\}$$

e)

$$\{(t_1, t_2), (t_1, t_3)\}$$

f)

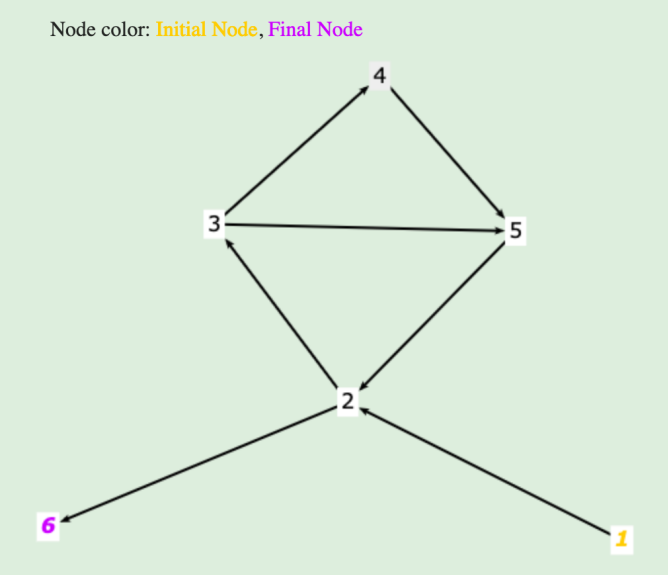
$$\{t_1, t_2, t_3\}$$

گراف ۴

**Graph IV.**

$N = \{1, 2, 3, 4, 5, 6\}$
$N_0 = \{1\}$
$N_f = \{6\}$
$E = \{(1,2), (2,3), (2,6), (3,4), (3,5), (4,5), (5,2)\}$
$def(1) = def(5) = use(5) = use(6) = \{x\}$
// Assume the use of x in 5 precedes the def
**Test Paths:**
$t1 = [1, 2, 6]$
$t2 = [1, 2, 3, 4, 5, 2, 3, 5, 2, 6]$
$t3 = [1, 2, 3, 5, 2, 3, 4, 5, 2, 6]$

a)




b)

1. [1, 2, 3, 4, 5] 2. [1, 2, 3, 5] 3. [1, 2, 6] 4. [5, 2, 3, 4, 5] 5. [5, 2, 3, 5] 6. [5, 2, 6]

c)

| | **direct** | **sidetrip** |
|---|---|---|
| **T1** | 3 | No |
| **T2** | 1, 5, 6 | No |
| **T3** | 2, 4, 6 | No |

d)

$$\{t_2\}, \{t_3\}$$

e)

$$\{t_1, t_2\}, \{t_1, t_3\}$$

f)

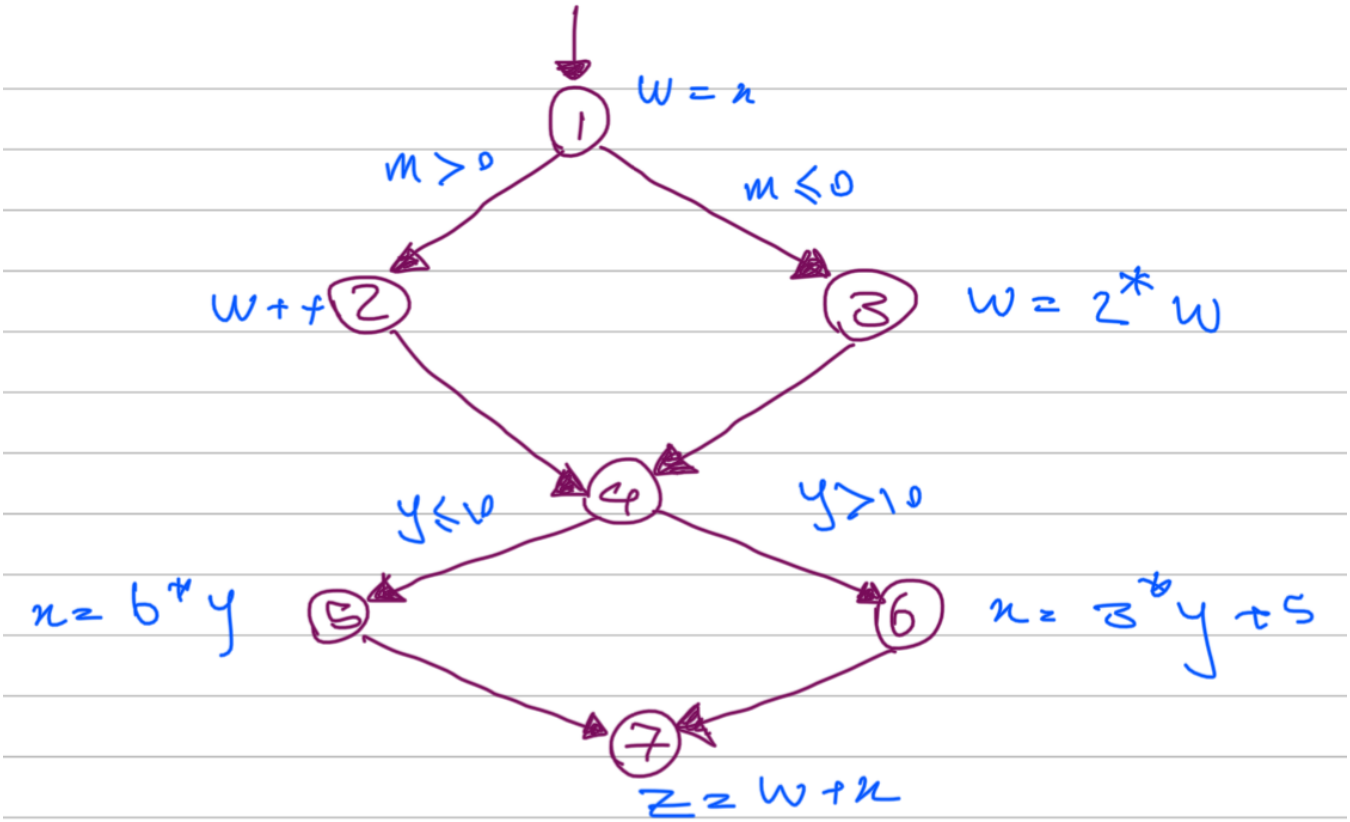
$$\{t_1, t_2, t_3\}$$

7.3

Q1.

Use the following program fragment for questions a–e below.

(a) Draw a control flow graph for this program fragment. Use the node numbers given above.




(b) Which nodes have defs for variable *w*?

1, 2, 3

(c) Which nodes have uses for variable *w*?

2, 3, 7

(d) Are there any du-paths with respect to variable *w* from node 1 to node 7? If not, explain why not. If any exist, show one.

در مسیرهای def-clear مرتبط باw ، از ۱ تا ۷ مسیری وجود ندارد. def که در گره‌های ۲ و ۳ وجود دارد، def موجود در گره ۱ را blcok می‌کند.

(e) List all of the du-paths for variables *w* and *x*.

| | | |
|---|---|---|
| **1** | [1, 2] | $W$ |
| **2** | [1, 3] | $W$ |
| **3** | [2, 4, 5, 7] | $W$ |
| **4** | [2, 4, 6, 7] | $W$ |
| **5** | [3, 4, 5, 7] | $W$ |
| **6** | [3, 4, 6, 7] | $W$ |
| **7** | [5, 7] | $X$ |
| **8** | [6, 7] | $X$ |

Q4.

Consider the pattern matching example in Figure 7.25. In particular, consider the final table of tests in Section 7.3. Consider the variable *iSub*. Number the (unique) test cases, starting at 1, from the top of the

*iSub* part of the table. For example, (*ab, c,* -1), which appears twice in

the *iSub* portion of the table, should be labeled test *t*4.
a) Give a minimal test set that satisfies *all defs* coverage. Use the test

cases given.

| | |
|---|---|
| **T1** | $(ab, ab, [])$ |
| **T2** | $(ab, a, 0)$ |
| **T3** | $(ab, ac, -1)$ |
| **T4** | $(ab, c, -1)$ |
| **T5** | $(a, bc, -1)$ |

| | |
|---|---|
| **T6** | $(abc, bc, 1)$ |
| **T7** | $(ab, b, 1)$ |
| **T8** | $(abc, ba, -1)$ |

برای پوشش همه دف‌ها، باید یکی از مسیرها را که با ۲ شروع می‌شود، و یکی از مسیرهای را که با ۱۰ شروع می‌شود، بگردیم. ۷ مجموعه مینیمال ممکن وجود دارد.

$$t_1, t_2, t_3, t_4, t_6, t_7, t_8$$

b) Give a minimal test set that satisfies *all uses* coverage.

$$\{t_1, t_4, t_5, t_6\}. \{t_1, t_5, t_6, t_7\}, \{t_1, t_5, t_6, t_8\}$$

c) Give a minimal test set that satisfies *all du-paths* coverage.

$$\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$$

Q8.

Consider the `equals()` method from the *java.util.AbstractList*

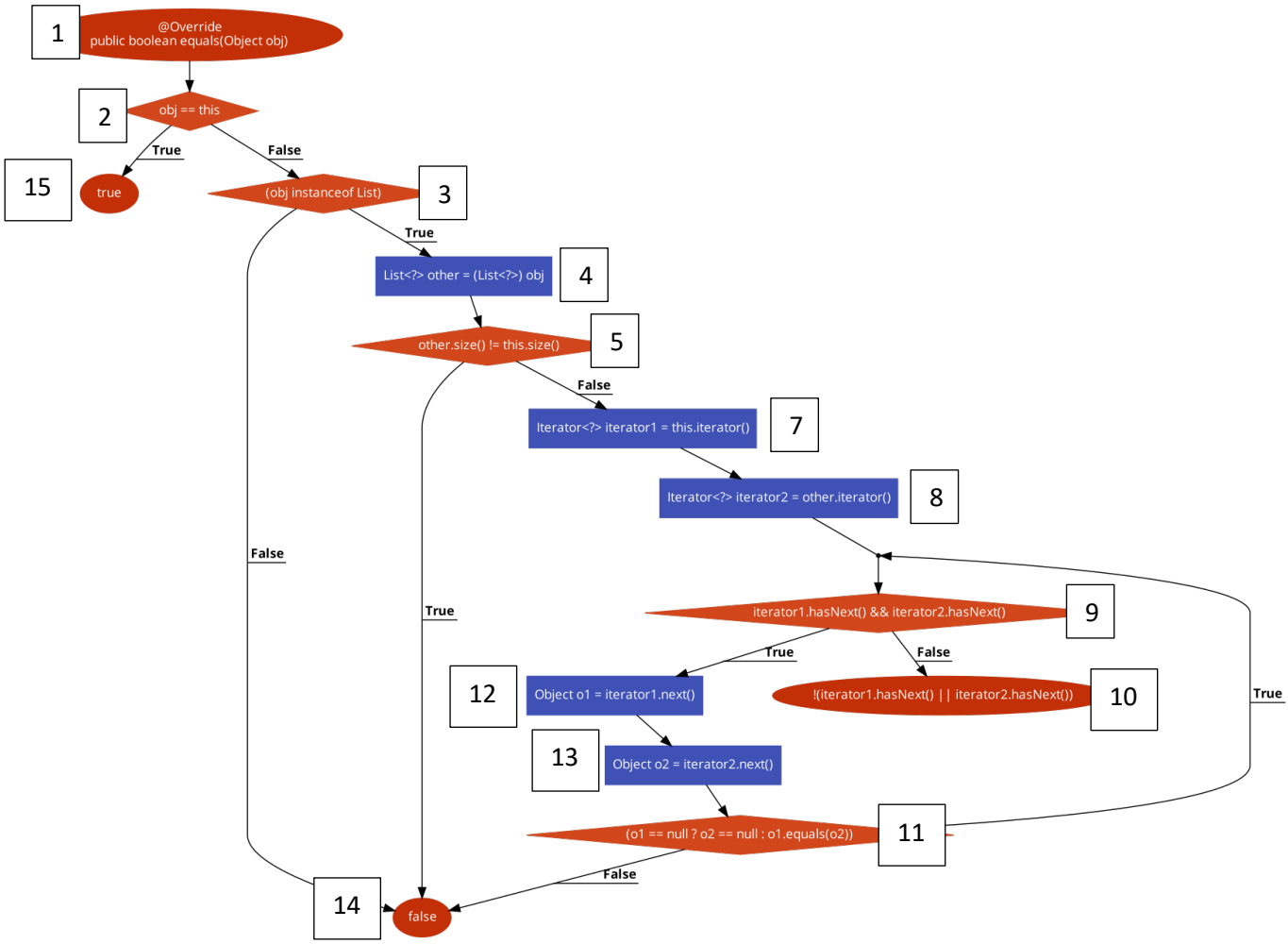
class:

```
@0verride
public boolean equals (0bject o)
{
  if (o == this) // A
     return true;
```

```
if (!(o instanceof List)) // B
  return false;

ListIterator e1 = listIterator();
ListIterator e2 = ((List) o).listIterator();
while (e1.hasNext() && e2.hasNext()) // C
{
  E o1 = e1.next();
  Object o2 = e2.next();
  if (!(o1 == null ? o2 == null : o1.equals (o2))) // D
    return false;
}
return !(e1.hasNext() || e2.hasNext()); // E
}
```

(a) Draw a control flow graph for this method. Several possible values can be used for the node number in the graph. Choose something reasonable.



(b) Label edges and nodes in the graph with the corresponding code fragments. You may abbreviate predicates as follows when labeling your graph:

A: `o == this`

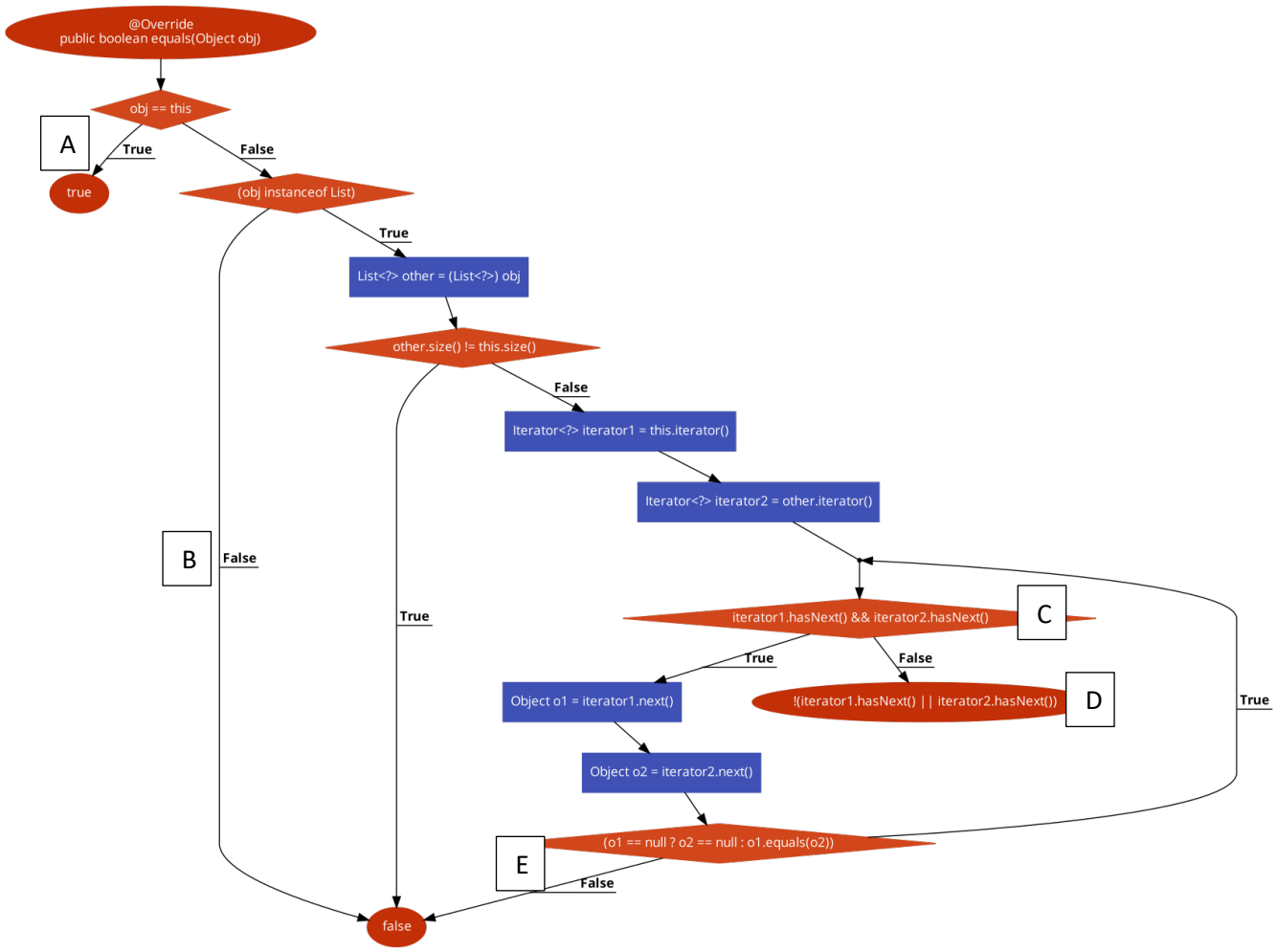
B: `!(o instanceof List)`

C: `e1.hasNext() && e2.hasNext()`

C: `e1.hasNext() && e2.hasNext()`

D: `!(o1 == null ? o2 == null : o1.equals(o2))`

E: `!(e1.hasNext() || e2.hasNext())`




(c) Node coverage requires (at least) four tests on this graph. Explain why.

گره ۱: این گره نشان دهنده شروع متد است. برای پوشاندن این گره، باید حداقل یک بار روش را اجرا کنید.

گره ۲: این گره مربوط به عبارت شرطی است که بررسی می کند آیا پارامتر obj همان نمونه لیست فعلی است (این). برای پوشاندن این گره، باید موردی را آزمایش کنید که در آن obj همان نمونه لیست است (این) و منجر به یک شرط واقعی می شود.

گره ۳: این گره بیانگر عبارت شرطی است که بررسی می کند آیا پارامتر obj نمونه ای از رابط List است یا خیر. برای پوشاندن این گره، باید موردی را آزمایش کنید که در آن obj نمونه ای از رابط List نیست و در نتیجه یک شرط نادرست ایجاد می شود.

گره ۴: این گره مربوط به مسیر اجرای باقی مانده پس از گذراندن دو بررسی شرطی اولیه است. این شامل تکرار دو لیست و مقایسه عناصر آنها برای برابری است. برای پوشش این گره، باید موردی را آزمایش کنید که در آن obj یک نمونه از رابط List است و با لیست فعلی متفاوت است (این). این امر مستلزم ارائه لیست های مختلف با عناصر یا اندازه های مختلف است.

## (d) Provide four tests (as calls to `equals()`) that satisfy node coverage on this graph. Make your tests short. You need to include output assertions. Assume that each test is independent and starts with the following state :

## Use the constants null, “ant”, “bat”, etc. as needed.

Test1:

```
List<String> list1 = new ArrayList<>(Arrays.asList("ant", "bat", "cat"));

List<String> list2 = null;


boolean result = list1.equals(list2);

System.out.println(result); // Expected output: false
```

توضیح: این تست مسیری را که در آن obj نمونه ای از واسط List نیست را پوشش می دهد. این یک نمونه لیست را با یک مرجع تهی مقایسه می کند، که باید false را برگرداند.

Test2:

```
List<String> list1 = new ArrayList<>(Arrays.asList("ant", "bat", "cat"));

List<String> list2 = new ArrayList<>(Arrays.asList("ant", "bat", "cat"));


boolean result = list1.equals(list2);

System.out.println(result); // Expected output: true
```

توضیح: این تست مسیری را پوشش می دهد که در آن هر دو لیست obj و فعلی نمونه هایی از رابط List هستند و دارای عناصر یکسان هستند. این دو لیست را با عناصر یکسان مقایسه می کند و در نتیجه درست است.

Test3:

```
List<String> list1 = new ArrayList<>(Arrays.asList("ant", "bat", "cat"));

List<Integer> list2 = new ArrayList<>(Arrays.asList(1, 2, 3));


boolean result = list1.equals(list2);

System.out.println(result); // Expected output: false
```

توضیح: این تست مسیری را پوشش می دهد که در آن هر دو لیست obj و فعلی نمونه هایی از رابط List هستند، اما انواع مختلفی دارند. این یک نمونه List<String> را با یک نمونه List<Integer> مقایسه می‌کند که نتیجه آن false است.

Test4:

```
List<String> list1 = new ArrayList<>(Arrays.asList("ant", "bat", "cat"));

List<String> list2 = new ArrayList<>(Arrays.asList("ant", "bat"));


boolean result = list1.equals(list2);

System.out.println(result); // Expected output: false
```

توضیح: این تست مسیری را پوشش می دهد که در آن هر دو لیست obj و فعلی نمونه هایی از رابط List هستند، اما اندازه های متفاوتی دارند. این دو لیست را با اندازه های مختلف مقایسه می کند که در نتیجه نادرست است.