

## ریپو گیت هاب


Repository: <https://github.com/elahekhodaverdi/SWT-Fall103>

Commit Hash: 4031f36c0e28b39ddab27e3f50dba9127d696349

## خروجی اجرای تست ها

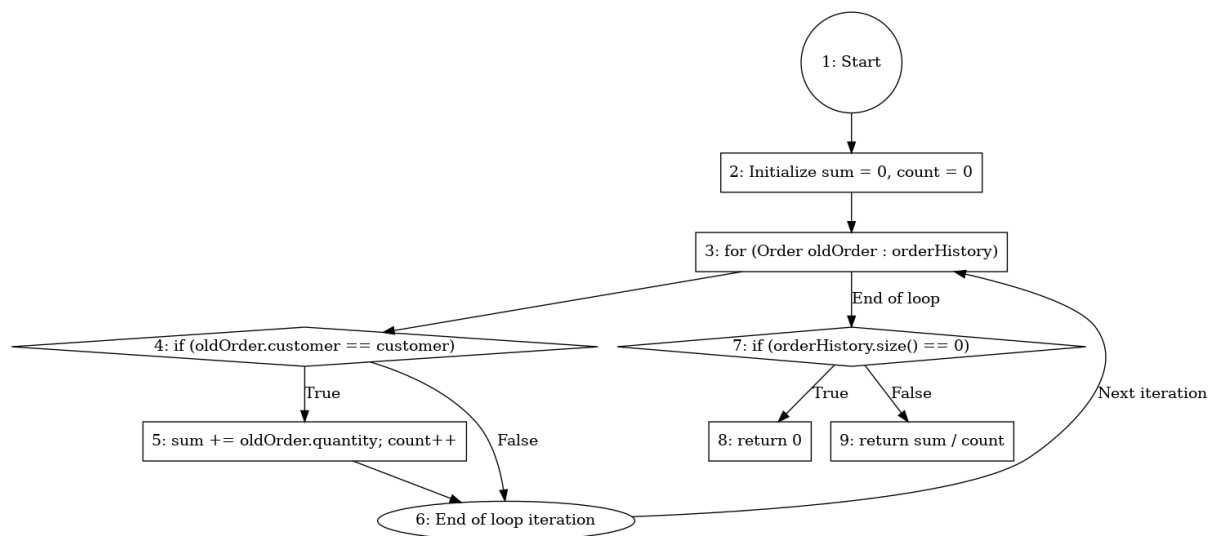
Fesadyab

Fesadyab

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 domain	<div><div></div></div>	100%	<div><div></div></div>	100%	0	22	0	45	0	7	0	2
Total	0 of 171	100%	0 of 30	100%	0	22	0	45	0	7	0	2

## سوال اول

(الف)



برای prime path ها داریم :

- A. [1,2,3,4,5,6]
- B. [4,5,6,3,7,8]
- C. [4,5,6,3,7,9]
- D. [3,4,5,6,3]

- E. [4,5,6,3,4]
- F. [1,2,3,7,9]
- G. [1,2,3,4,6]
- H. [1,2,3,7,8]
- I. [6,3,4,5,6]
- J. [5,6,3,4,5]
- K. [4,6,3,7,8]
- L. [4,6,3,7,9]
- M. [3,4,6,3]
- N. [6,3,4,6]
- O. [4,6,3,4]

در این مسیر ها سه مسیر [4,5,6,3,7,8] و [1,2,3,7,9] و [4,6,3,7,8] طبق لاجیک برنامه infeasible هستند.

برای du path ها داریم:

[1,2,3,4]	customer
[5,6,3,7,9], [5,6,3,4,5], [2,3,7,9], [2,3,4,5]	sum
[5,6,3,7,9], [5,6,3,4,5], [2,3,7,9], [2,3,4,5]	count
[3,4,5], [3,4]	oldOrder
[1,2,3,7], [1,2,3]	orderHistory

در اینجا هم برای sum و count مسیر [2,3,7,9] مسیری infeasible است.  
مسیرهای نهایی بعد حذف duplication ها:

- A. [1, 2, 3, 4]
- B. [2,3,4,5]
- C. [2,3,7,9]
- D. [5,6,3,4,5]
- E. [5,6,3,7,9]
- F. [3,4]

G. [3,4,5]

H. [1,2,3]

I. [1,2,3,7]

(ب)

برای کاور کردن prime path ها به تست‌های زیر نیاز است:

Prime path	
Test path	TRs toured
[1,2,3,4,5,6,3,4,6,3,7,9]	A, D, E, L, M, N
[1,2,3,7,8]	<b>H</b>
[1,2,3,4,6,3,4,5,6,3,7,9]	C, D, G, I, O
[1,2,3,4,5,6,3,4,5,6,3,7,9]	A, C, D, E, I, <b>J</b>
[1,2,3,4,6,3,4,6,3,7,9]	G, L, M, N, O

علاوه بر تست‌های بالا سه تست [1,2,3,4,5,6,3,7,8] و [1,2,3,4,6,3,7,8] و [1,2,3,7,9] هم نیاز بود تا تمام prime path ها کاور شوند ولی چون برای کاور کردن مسیرهای infeasible استفاده می‌شدند از نوشتنشان صرف نظر شد.

برای کاور کردن du path ها به تست‌های زیر نیاز است:

Du path	
Test path	TRs toured
[1,2,3,7,8]	<b>I</b> , H
[1,2,3,4,5,6,3,4,5,6,3,7,9]	<b>A, B, D, E, F, G</b> , H

علاوه بر تست‌های بالا تست [1,2,3,7,9] هم نیاز بود تا تمام du path ها کاور شوند ولی چون برای کاور کردن مسیرهای infeasible استفاده می‌شد از نوشتن آن صرف نظر شد.

## (ج)

در حین اجرای تست [1,2,3,4,6,3,4,6,3,7,9] ما در history سفارشات خود دو عدد سفارش داریم که هیچ یک برای مشتری ما نیست پس بعد از دوبار لوپ زدن روی آرایه ما انتظار داریم چون این سفارش وجود نداشته مقدار صفر را به عنوان خروجی دریافت کنیم اما چون داریم دو صفر را بر هم تقسیم می‌کنیم exception خواهیم گرفت.  
برای رفع آن داریم باید این قسمت کد :

```
if (orderHistory.size() == 0) { return 0; }
```

با این کد جایگزین شود:

```
if (count == 0) { return 0; }
```

## سوال دوم

## (الف)

Branch coverage: حداقل سه آزمایش مورد نیاز است.

1. a = true, b = false, c = false
2. a = false, b = false, c = true
3. a = false, b = false, c = false

Statement coverage: حداقل دو آزمایش مورد نیاز است.

1. a = true, b = false, c = false
2. a = false, b = false, c = true

## (ب)

بله می‌شود.

می‌توان تمام بدنه تابع را به فرم زیر تبدیل کرد:

```
var1, var2 = false, false;
var1 = (a | b);
var2 = ~(a | b) & c;
return var1, var2
```

(ج)

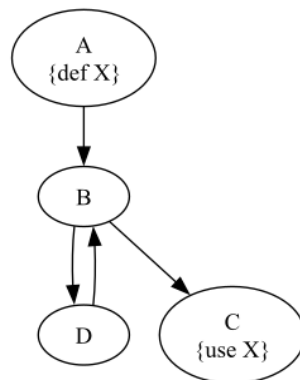
یک شرط کافی در cfg توابع برای اینکه به پوشش 100% برسند این است که یک مسیر وجود داشته باشد که در آن تمام نودها را ویزیت کرده باشیم.

برای مثال براساس cfg توابع، برای اینکه بتوان با یک تست به 100% statement coverage رسید نباید شاخه‌ای در برنامه وجود داشته باشد که گراف را به دو زیرگراف تقسیم کند که هیچ مسیری به هم ندارند. منظور از شاخه وجود if و else و for و while, switch و موارد مشابه است که حالت شرطی به وجود می‌آورند و براساس آن‌ها شاخه‌های مختلفی برای ادامه اجرای برنامه خواهیم داشت. هرچند وجود این موارد به خودی خود ممکن است به طور قطعی مانع نباشند. گاه وجود return در آن‌ها مانع پوشش 100 درصدی می‌شود زیرا باعث ایجاد دو زیر گراف کاملاً جدا خواهد شد که به یکدیگر مسیری ندارند. هرچند موارد دیگری مانند unreachable node ها نیز وجود دارد که باعث ایجاد مشکل در پوشش می‌شوند، این موارد هم باید در کد اصلاح شوند هم اینکه در cfg این node ها می‌توانند حذف شوند و در cfg تاثیری ندارند و نکته اصلی وجود مسیری است که از تمام نودها گذر می‌کند.

## سوال سوم

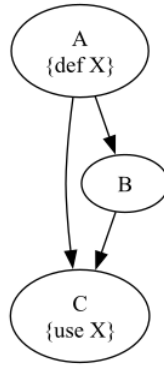
از برهان خلف کمک می‌گیریم. فرض می‌کنیم که با پوشش مسیرهای du مسیرهای prime نیز کاور می‌شوند.

حال گراف زیر را در نظر بگیرید:



در این گراف اگر بخواهیم du path ها را کاور کنیم کافی است که  $A \rightarrow B \rightarrow C$  را در نظر بگیریم. این در حالی است که ما برای مثال prime path مربوط به  $B \rightarrow D \rightarrow B$  را در نظر نمی‌گیریم و کاور نمی‌کنیم. پس فرضیه را نقض می‌کند.

یک مثال دیگر گراف زیر است:



در این گراف نیز برای کاور کردن du path ها کفایت یک تست برای  $A \rightarrow C$  بنویسیم اما دو prime path داریم که برای کاور کردن آنها به دو تست نیاز داریم و با کاور کردن du path ها تنها یکی از آنها را کاور کرده‌ایم.

پس فرض خلف اشتباه بوده حکم ثابت می‌شود.