Non deterministic path analysis on cFlow

WEN FAN

2021.9

Content

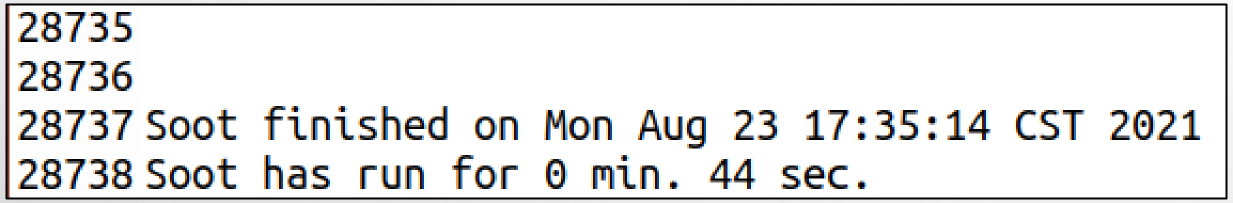
- Problem
- Reason
- Solution
- Result
- Next step

Content

- Problem
- Reason
- Solution
- Result
- Next step

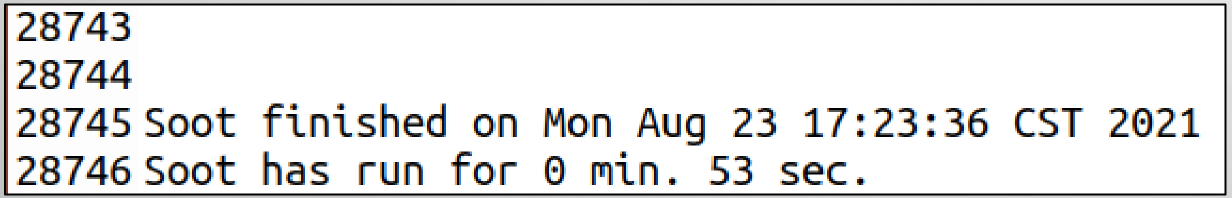
Problem

- Run cFlow on hadoop_common 3.3.0 several times,
- And I get two different outputs.



```
28735
28736
28737 Soot finished on Mon Aug 23 17:35:14 CST 2021
28738 Soot has run for 0 min. 44 sec.
```

a.txt: 28738 lines

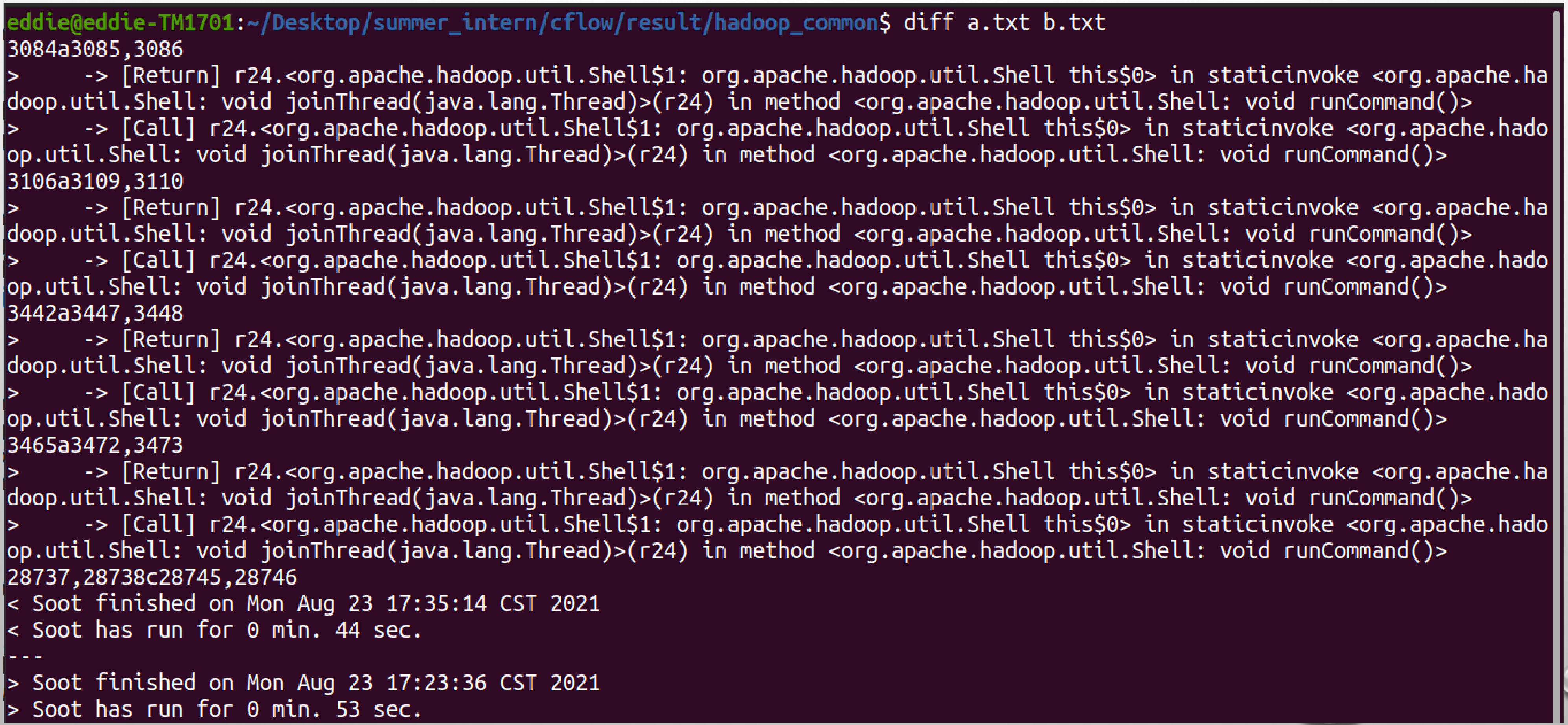


```
28743
28744
28745 Soot finished on Mon Aug 23 17:23:36 CST 2021
28746 Soot has run for 0 min. 53 sec.
```

b.txt: 28746 lines

Problem

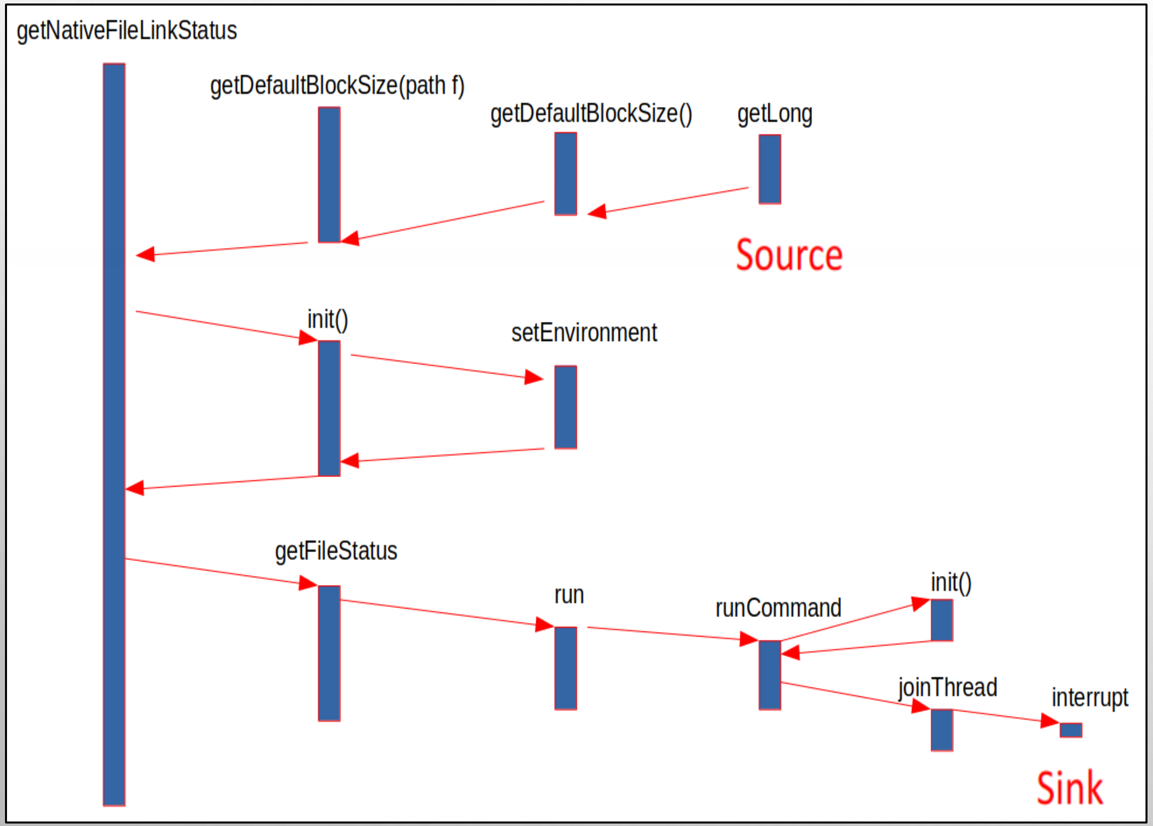
- b.txt has more taints about call and return on method joinThread().



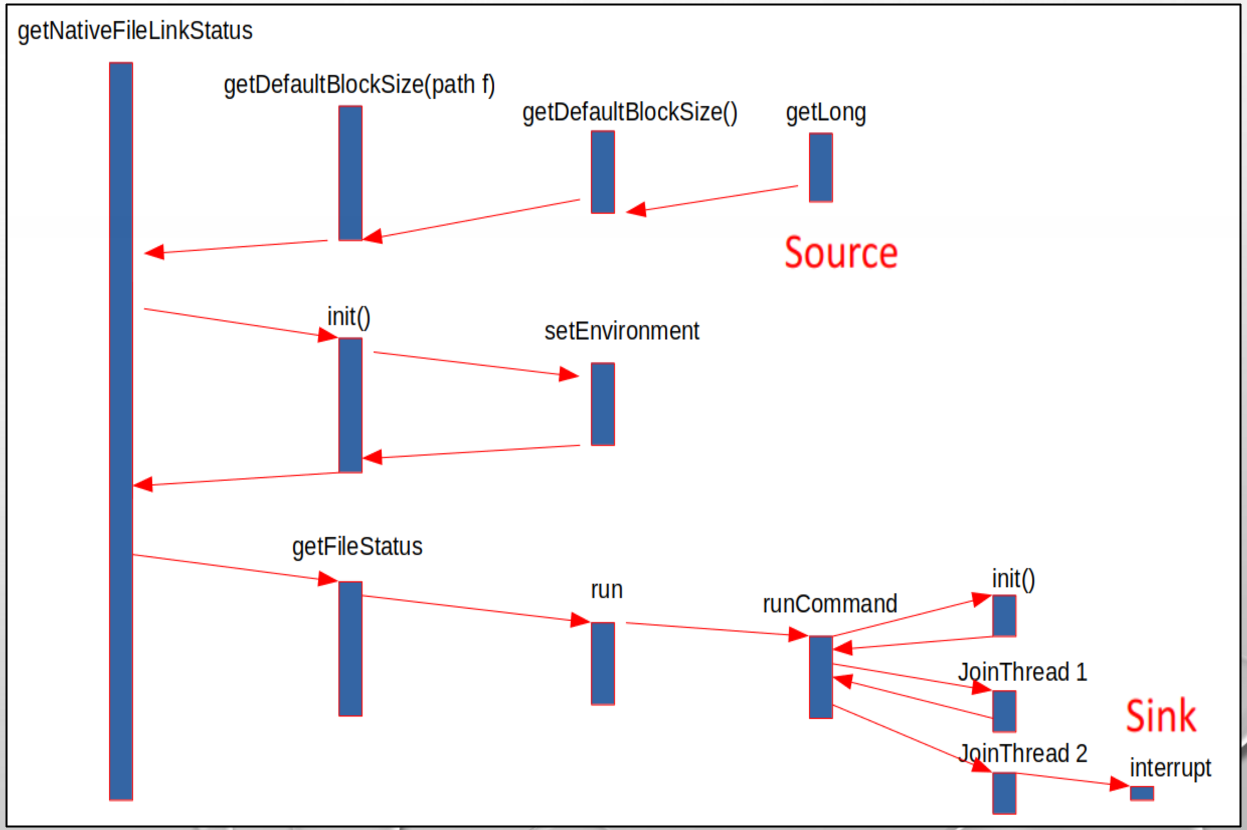
```
eddie@eddie-TM1701:~/Desktop/summer_intern/cflow/result/hadoop_common$ diff a.txt b.txt
3084a3085,3086
>     -> [Return] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
>     -> [Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
3106a3109,3110
>     -> [Return] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
>     -> [Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
3442a3447,3448
>     -> [Return] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
>     -> [Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
3465a3472,3473
>     -> [Return] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
>     -> [Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24) in method <org.apache.hadoop.util.Shell: void runCommand()>
28737,28738c28745,28746
< Soot finished on Mon Aug 23 17:35:14 CST 2021
< Soot has run for 0 min. 44 sec.
---
> Soot finished on Mon Aug 23 17:23:36 CST 2021
> Soot has run for 0 min. 53 sec.
```

Problem

- I trace the taint propagation path(at line 3084) that contains the difference above.



a.txt



b.txt

Content

- Problem
- Reason
- Solution
- Result
- Next step

Reason

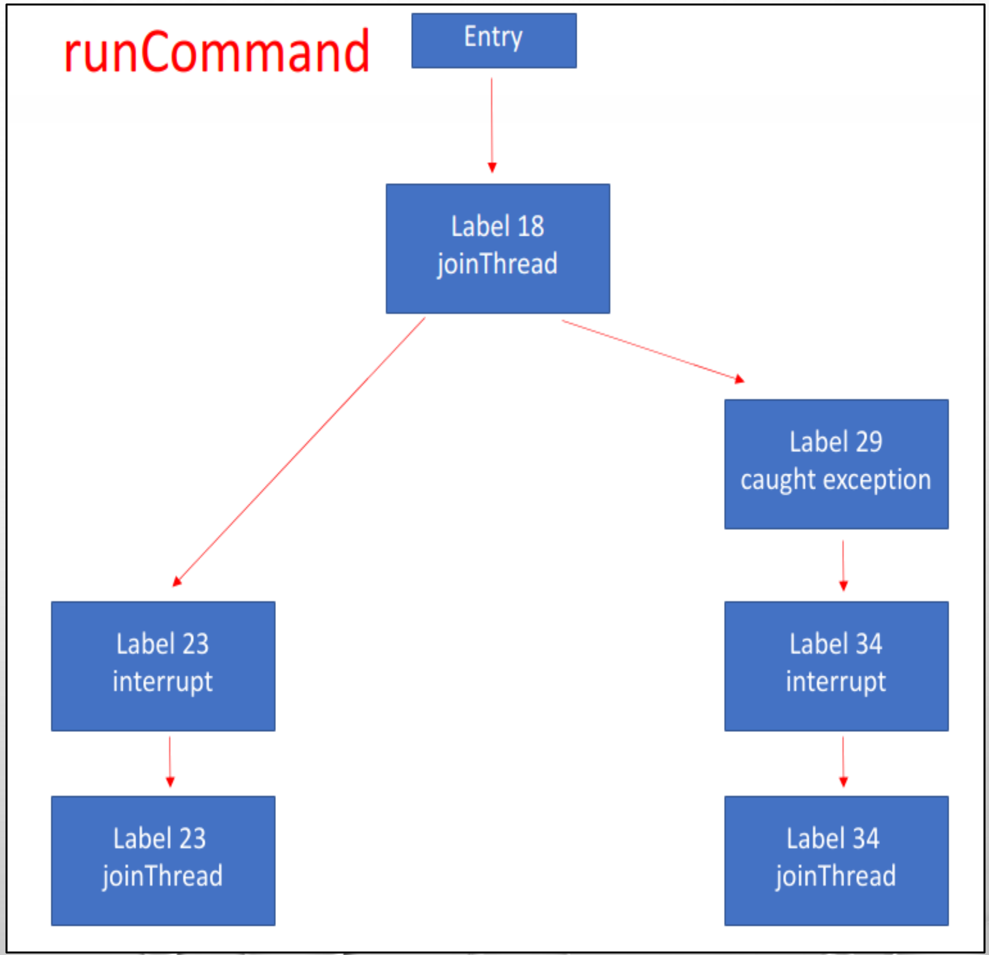
- For simplicity, I adjust cFlow by
  - Printing all taint propagation paths between each pair of source and sink.
  - Only detecting method getLong() as source and method interrupt() as sink.
  - Only Considering the taint propagation path after method runCommand().
- Also, I analyze the control flow graph of method runCommand() and joinThread().

Reason

- Here is the logic structure of method runCommand().

```
private void runCommand() throws IOException {
    ...
    try {
        ...
        joinThread(errThread);
        completed.set(true);
        ...
    } catch (InterruptedException ie) {
        ...
    } finally {
        ...
        if (!completed.get()) {
            errThread.interrupt();
            joinThread(errThread);
        }
        ...
    }
}
```

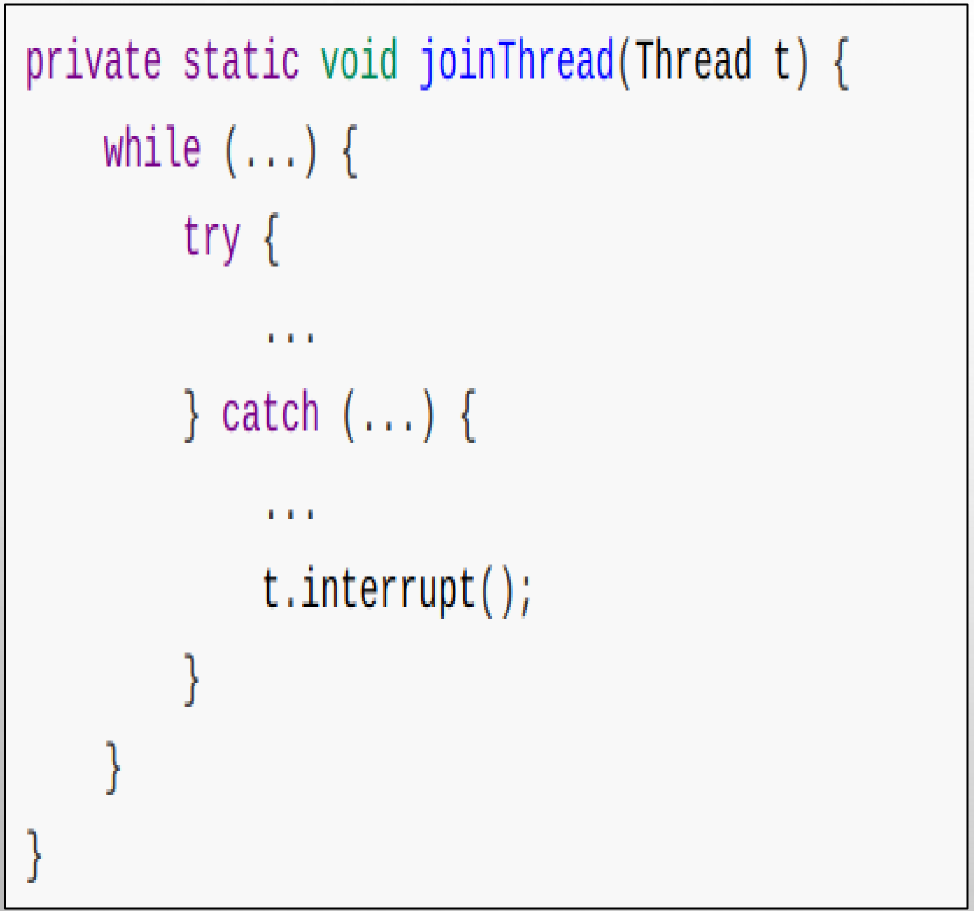
Source code



CFG

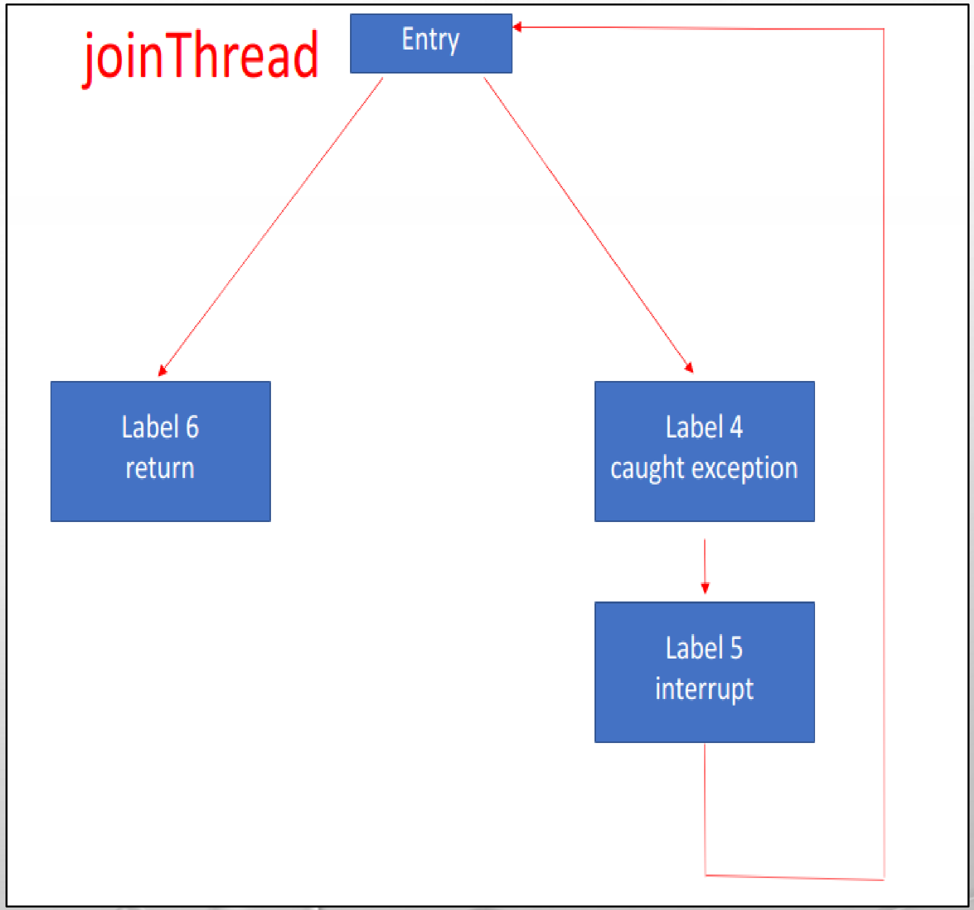
Reason

- Here is the logic structure of method joinThread().



```
private static void joinThread(Thread t) {
    while (...) {
        try {
            ...
        } catch (...) {
            ...
            t.interrupt();
        }
    }
}
```

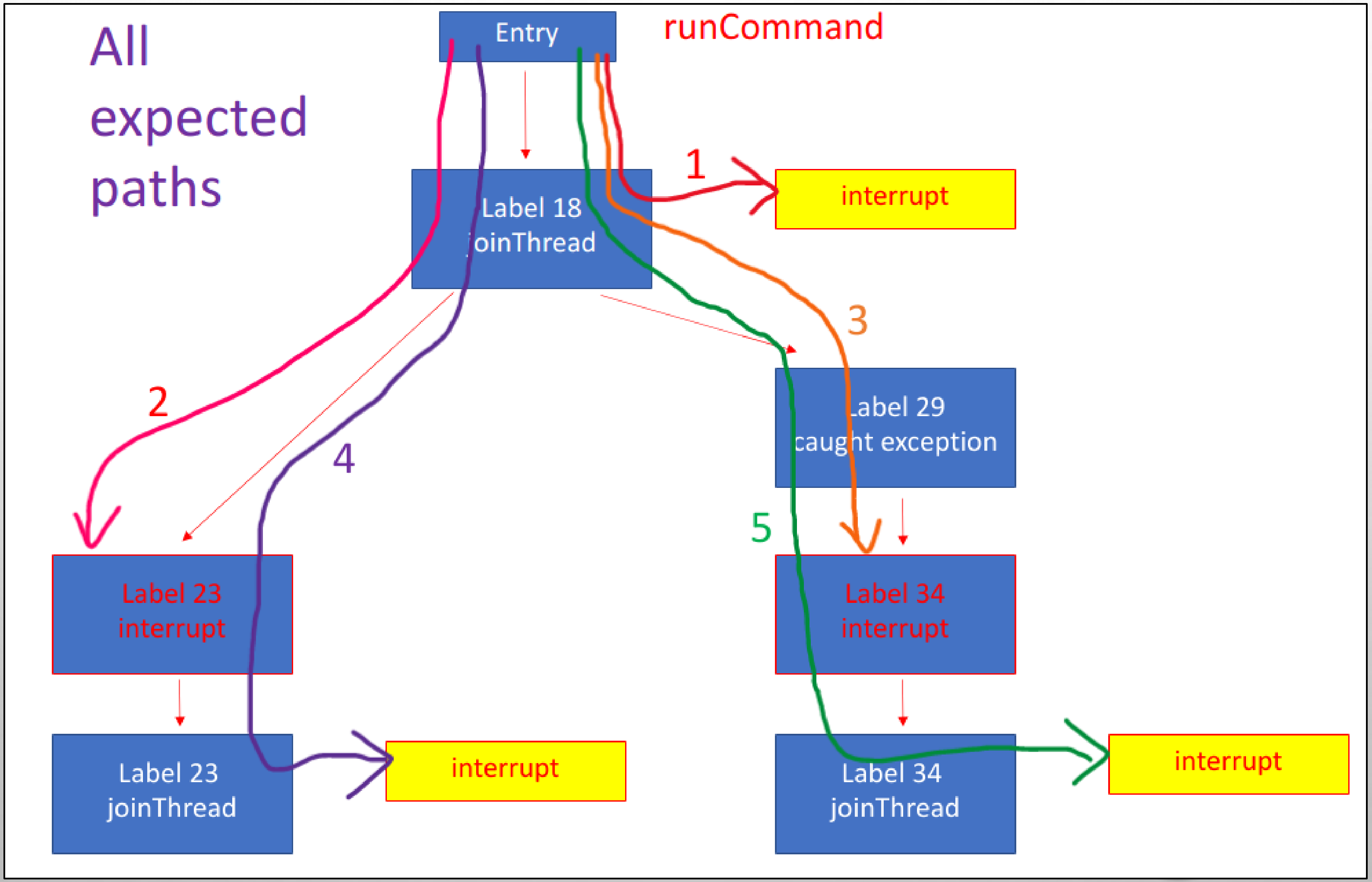
Source code



CFG

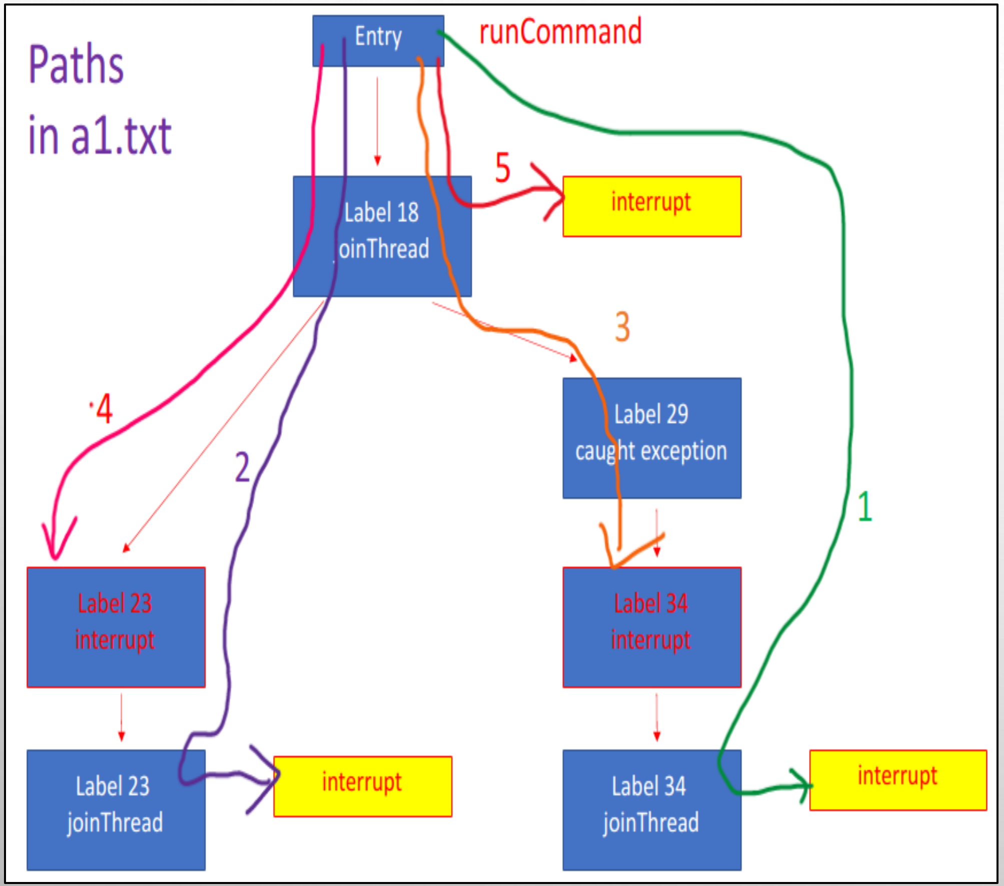
Reason

- Theoratically, there are five propagation paths to interrupt().

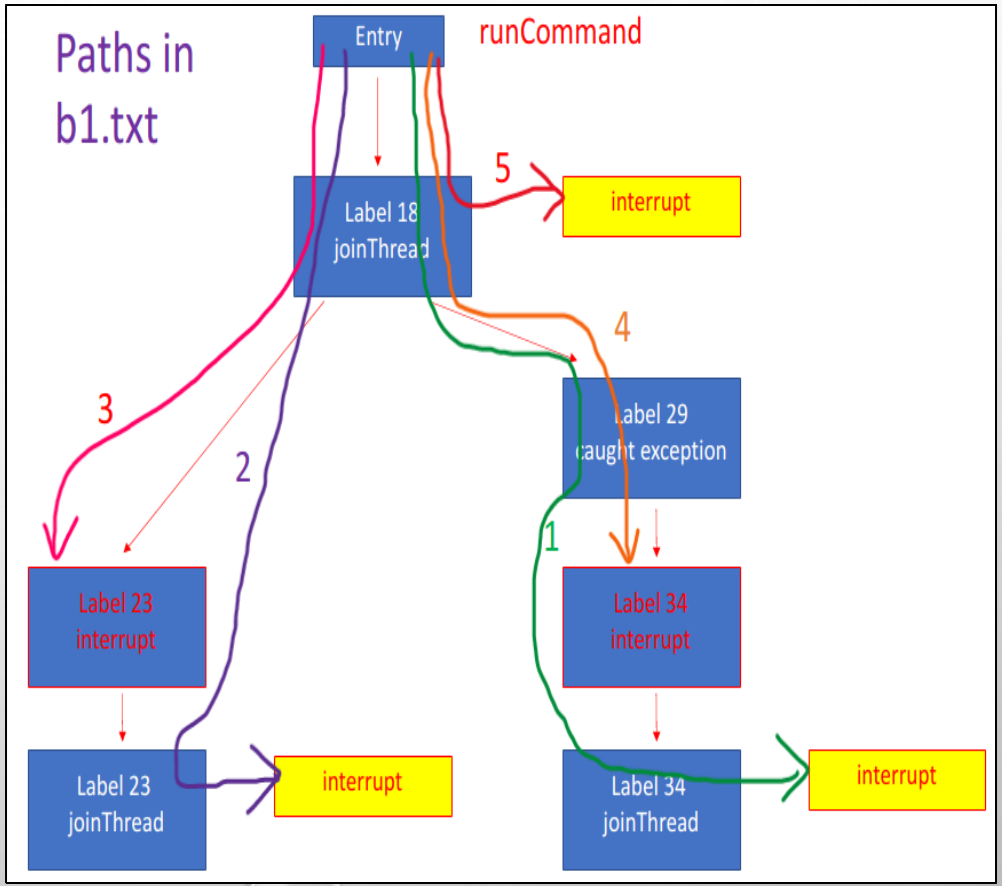


Reason

- However, I get two different outputs after this adjustment.



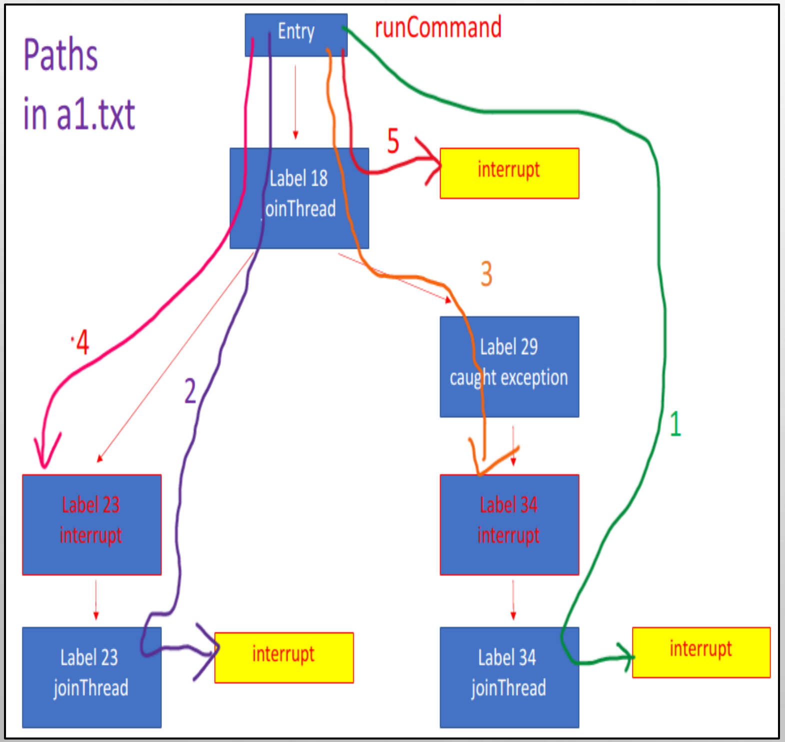
a1.txt



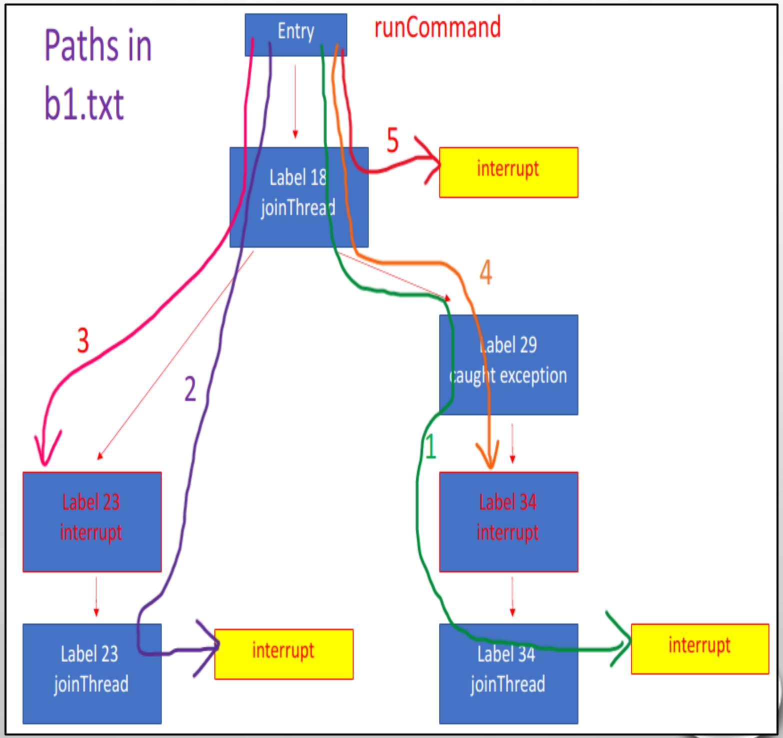
b1.txt

Reason

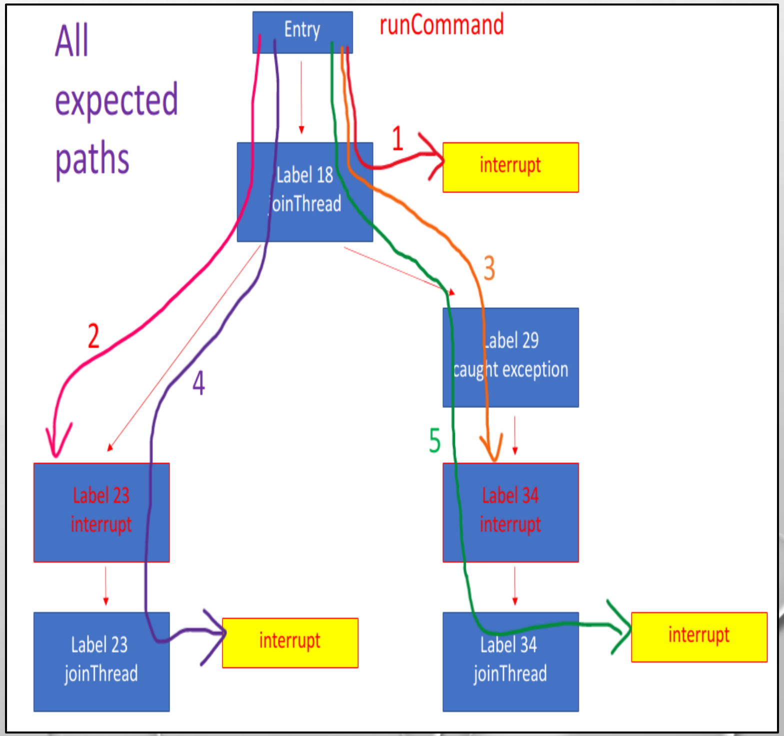
- From a1.txt and b1.txt, we can find that
  - joinThread() at Label 23 and Label 34 can be reached by skipping interrupt().
  - only path 1 in a1.txt and path 1 in b1.txt are different.



a1.txt



b1.txt



expected

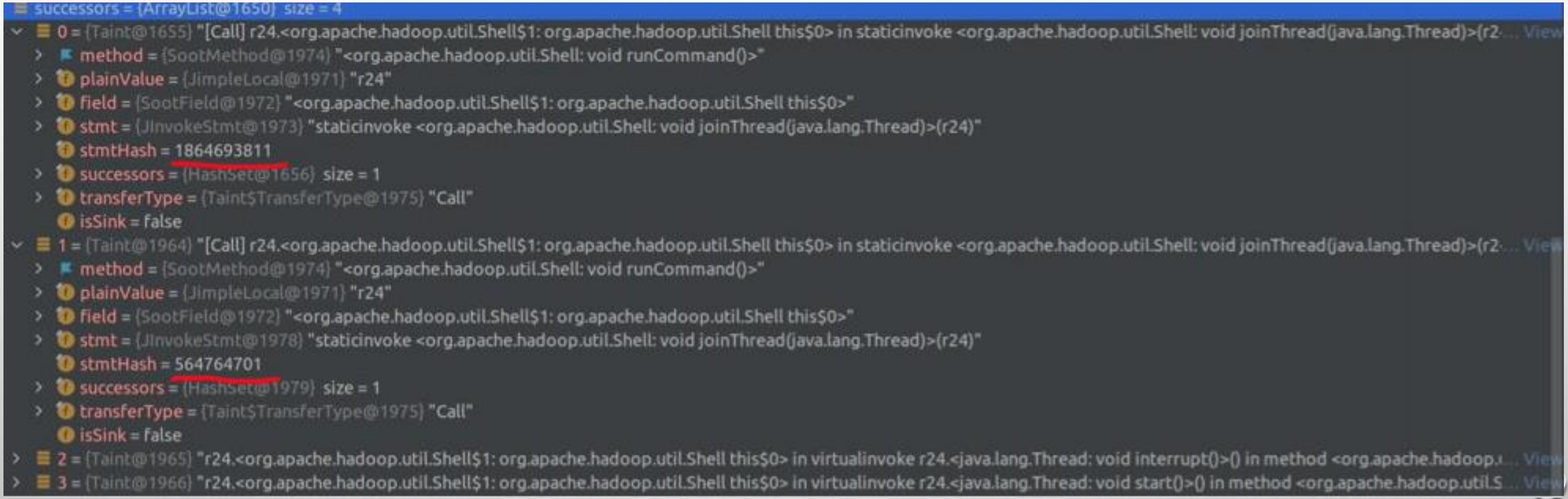
Reason

- I use IntelliJ IDEA to trace the path reconstruction step.
- Before the trace, I record the identityHashCode of each invoke statement of method joinThread() and interrupt().
- I have found a difference when current taint is

```
r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in r24
= $r87 in method <org.apache.hadoop.util.Shell: void runCommand()>
```

Reason

- In one case, the sorted list successors is

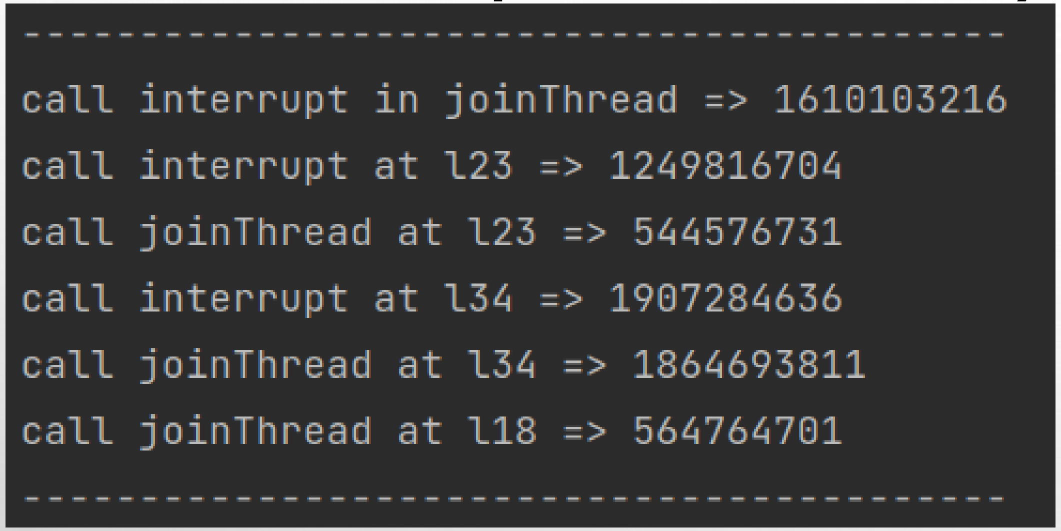


```
≡ successors = {ArrayList@1650}  size = 4
∨ ≡ 0 = {Taint@1655} "[Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r2... View
  > method = {SootMethod@1974} "<org.apache.hadoop.util.Shell: void runCommand()>"
  > plainValue = {JimpleLocal@1971} "r24"
  > field = {SootField@1972} "<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0>"
  > stmt = {JInvokeStmt@1973} "staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24)"
    stmtHash = 1864693811
  > successors = {HashSet@1656}  size = 1
  > transferType = {Taint$TransferType@1975} "Call"
    isSink = false
∨ ≡ 1 = {Taint@1964} "[Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r2... View
  > method = {SootMethod@1974} "<org.apache.hadoop.util.Shell: void runCommand()>"
  > plainValue = {JimpleLocal@1971} "r24"
  > field = {SootField@1972} "<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0>"
  > stmt = {JInvokeStmt@1978} "staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24)"
    stmtHash = 564764701
  > successors = {HashSet@1979}  size = 1
  > transferType = {Taint$TransferType@1975} "Call"
    isSink = false
> ≡ 2 = {Taint@1965} "r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in virtualinvoke r24.<java.lang.Thread: void interrupt()>() in method <org.apache.hadoop.... View
> ≡ 3 = {Taint@1966} "r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in virtualinvoke r24.<java.lang.Thread: void start()>() in method <org.apache.hadoop.util.S... View
```

{ Invoke joinThread 1864693811 | Invoke joinThread 564764701 | Invoke interrupt | Invoke start }

Reason

- Since the Map from IdentityHashCode to Statement is



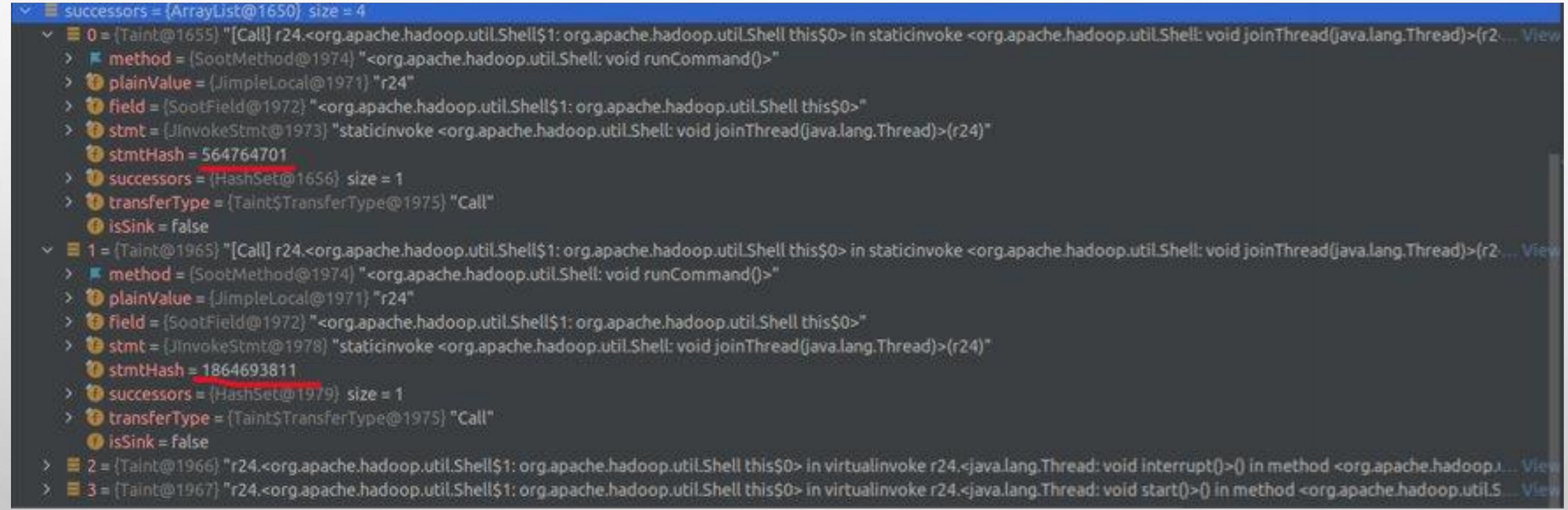
```
-----------------------------------------
call interrupt in joinThread => 1610103216
call interrupt at l23 => 1249816704
call joinThread at l23 => 544576731
call interrupt at l34 => 1907284636
call joinThread at l34 => 1864693811
call joinThread at l18 => 564764701
-----------------------------------------
```

- So successors is

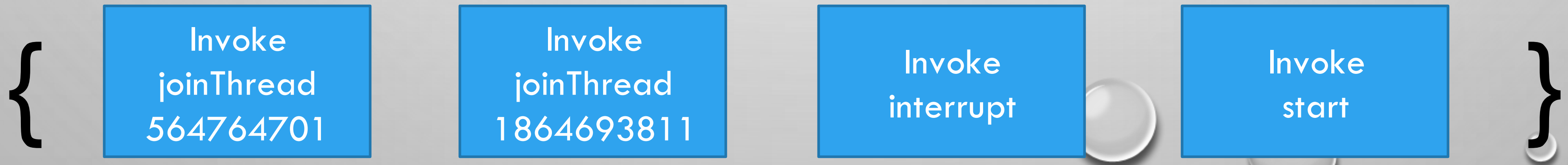
{ Invoke joinThread at Label 34 | Invoke joinThread at Label 18 | Invoke interrupt | Invoke start }

Reason

- In another case, the sorted list successors is

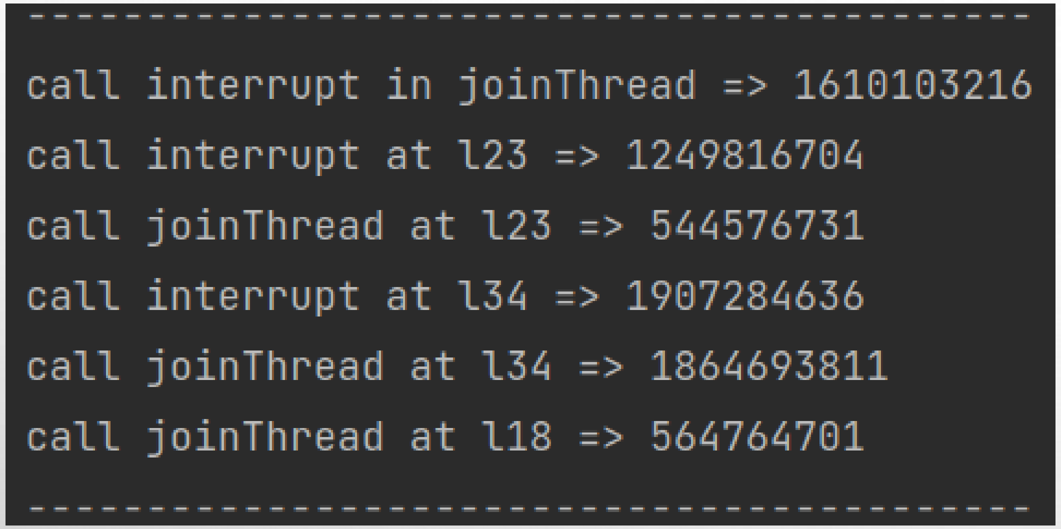


```
successors = {ArrayList@1650}  size = 4
  0 = {Taint@1655} "[Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r2... View
    method = {SootMethod@1974} "<org.apache.hadoop.util.Shell: void runCommand()>"
    plainValue = {JimpleLocal@1971} "r24"
    field = {SootField@1972} "<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0>"
    stmt = {JInvokeStmt@1973} "staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24)"
    stmtHash = 564764701
    successors = {HashSet@1656}  size = 1
    transferType = {Taint$TransferType@1975} "Call"
    isSink = false
  1 = {Taint@1965} "[Call] r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r2... View
    method = {SootMethod@1974} "<org.apache.hadoop.util.Shell: void runCommand()>"
    plainValue = {JimpleLocal@1971} "r24"
    field = {SootField@1972} "<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0>"
    stmt = {JInvokeStmt@1978} "staticinvoke <org.apache.hadoop.util.Shell: void joinThread(java.lang.Thread)>(r24)"
    stmtHash = 1864693811
    successors = {HashSet@1979}  size = 1
    transferType = {Taint$TransferType@1975} "Call"
    isSink = false
  2 = {Taint@1966} "r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in virtualinvoke r24.<java.lang.Thread: void interrupt()>() in method <org.apache.hadoop.... View
  3 = {Taint@1967} "r24.<org.apache.hadoop.util.Shell$1: org.apache.hadoop.util.Shell this$0> in virtualinvoke r24.<java.lang.Thread: void start()>() in method <org.apache.hadoop.util.S... View
```



Reason

- Since the Map from IdentityHashCode to Statement is



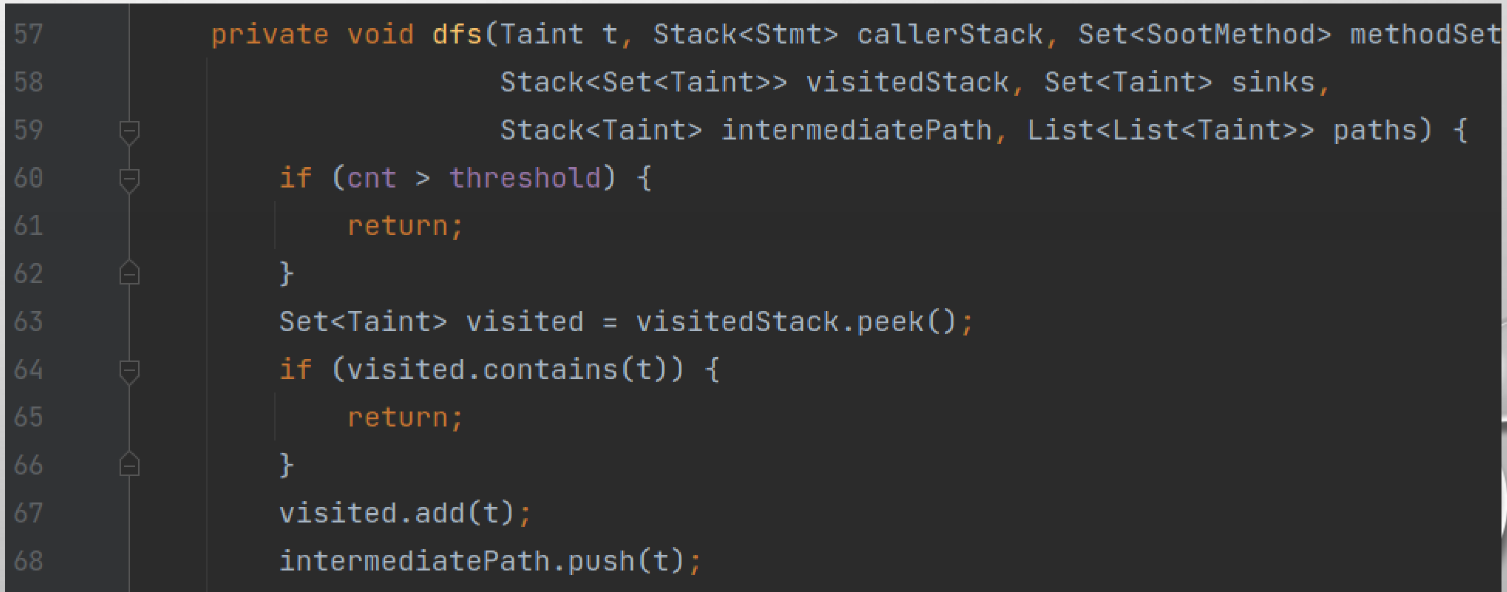
```
------------------------------------------

call interrupt in joinThread => 1610103216
call interrupt at l23 => 1249816704
call joinThread at l23 => 544576731
call interrupt at l34 => 1907284636
call joinThread at l34 => 1864693811
call joinThread at l18 => 564764701

------------------------------------------
```

- So successors is

{ Invoke joinThread at Label 18 | Invoke joinThread at Label 34 | Invoke interrupt | Invoke start }

Reason

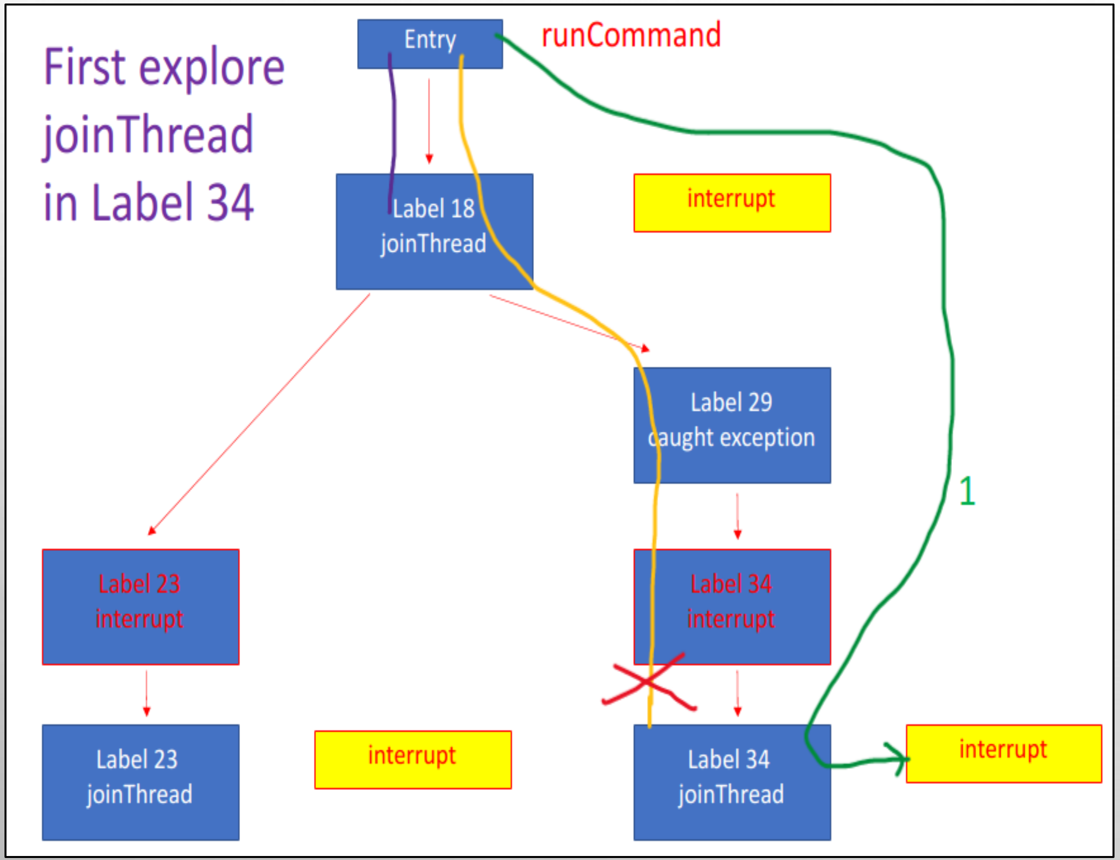
- Different sequence in successors(as an open list) can cause different sequence of DFS searching, which contributes to different paths due to visitedStack.
- VisitedStack can avoid repetitious visit to one taint in the same procedure.



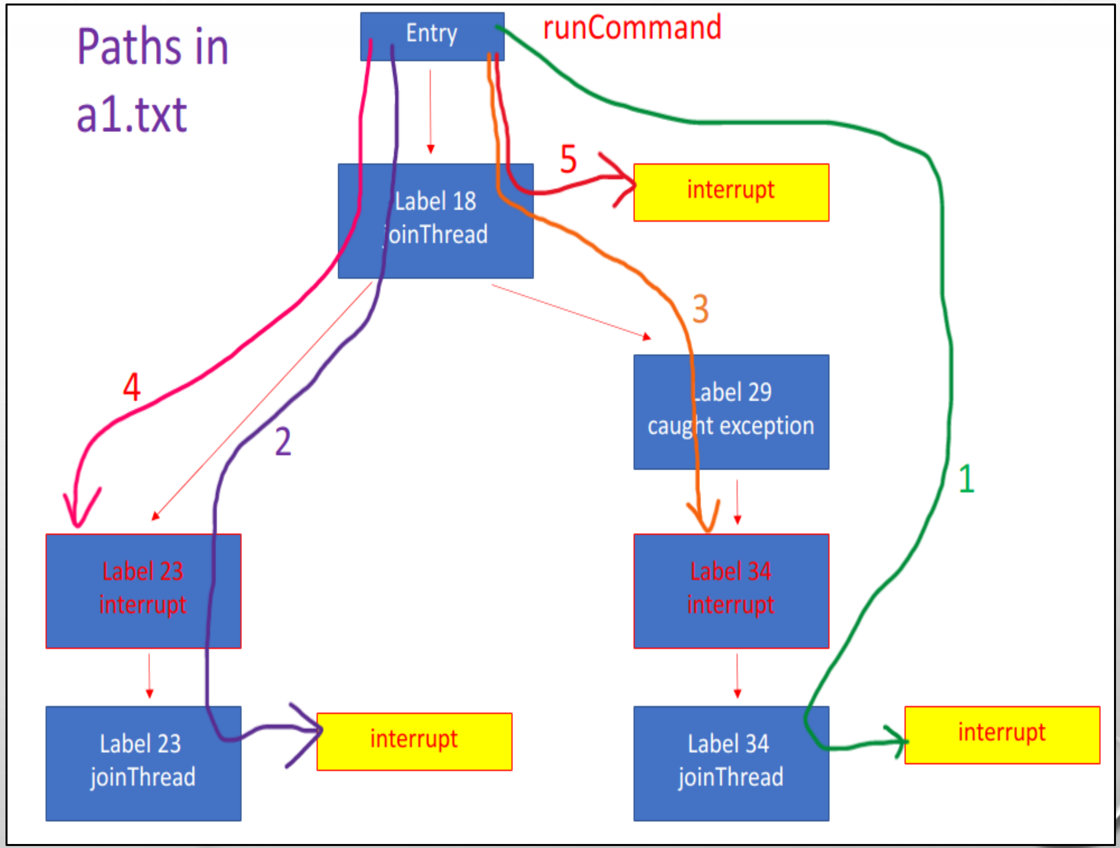
```
    private void dfs(Taint t, Stack<Stmt> callerStack, Set<SootMethod> methodSet
                     Stack<Set<Taint>> visitedStack, Set<Taint> sinks,
                     Stack<Taint> intermediatePath, List<List<Taint>> paths) {
        if (cnt > threshold) {
            return;
        }
        Set<Taint> visited = visitedStack.peek();
        if (visited.contains(t)) {
            return;
        }
        visited.add(t);
        intermediatePath.push(t);
```

Reason

- The first case causes the result of a1.txt.



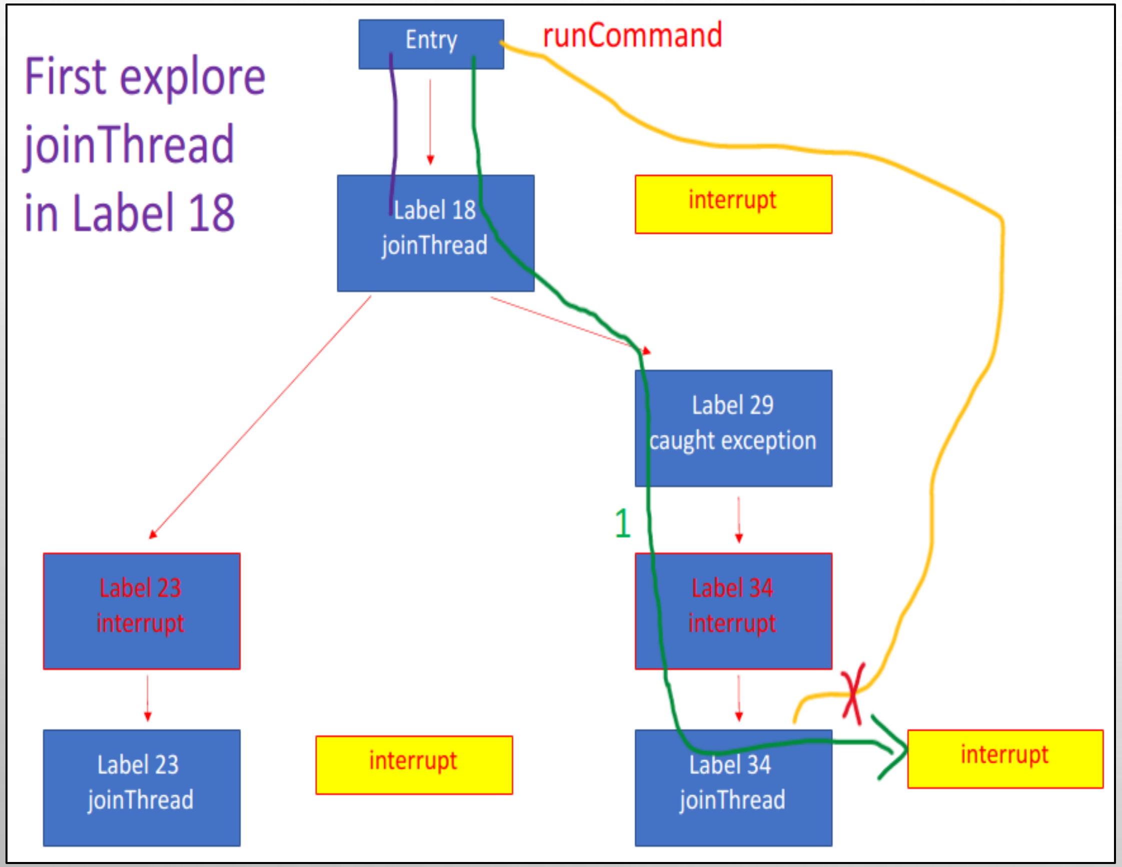
Searching case 1



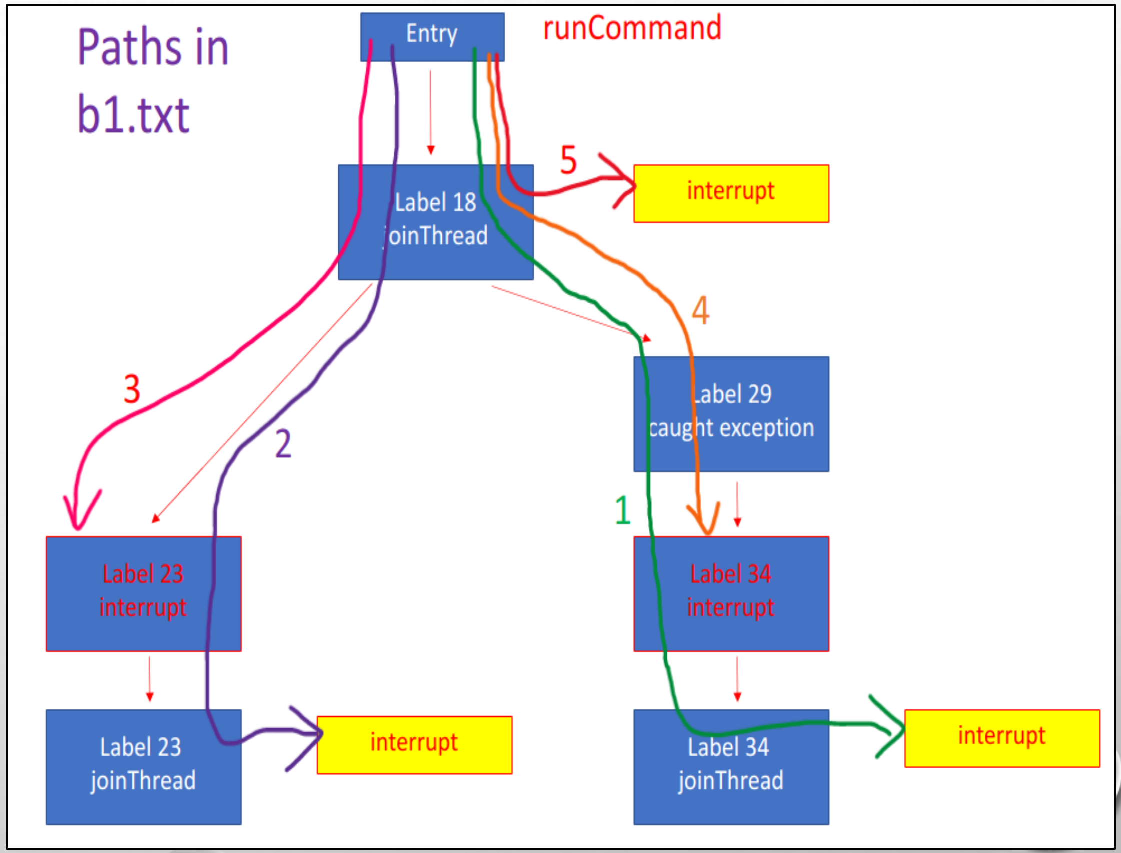
a1.txt

Reason

- Another case causes the result of b1.txt.



Searching case 2



b1.txt

Reason

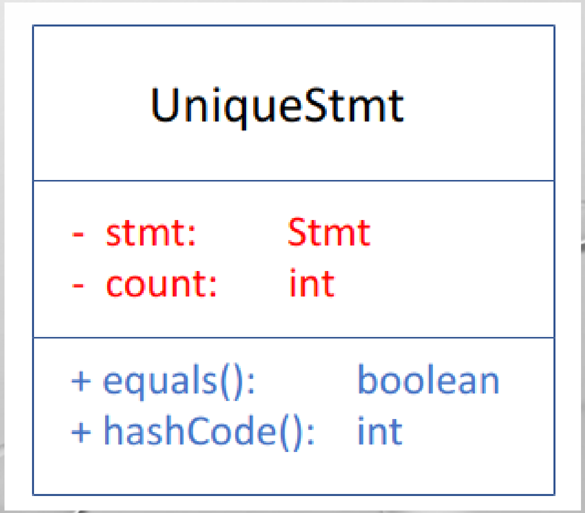
- Why successors is not deterministically sorted after the sort?
  - Originally, the sequence of taints appended into successors is not deterministic(Soot???)
  - The sort is stable and it is based on the comparison on the string representation of taint. However, Two taints on invoke statement of joinThread() has the same string representation

Content

- Problem
- Reason
- Solution
- Result
- Next step

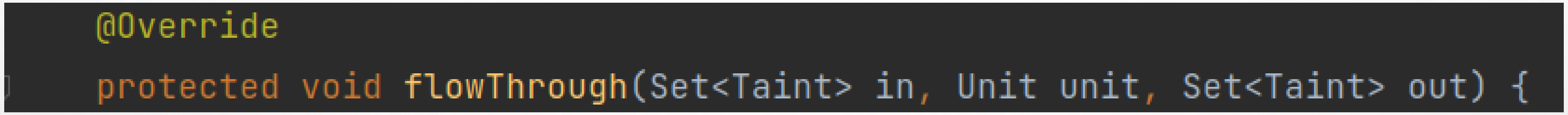
Solution

- Since it is not easy to change the feature of Soot , I try to add more information to each statement for comparison.
- Here, I encapsulate each statement with its sequence number(count) as UniqueStmt, so those statements are distinct in comparison.



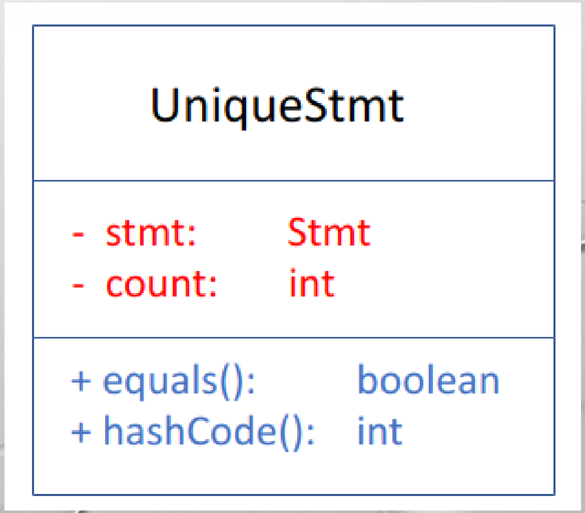
Solution

- I get UniqueStmt for each statement in method flowThrough()



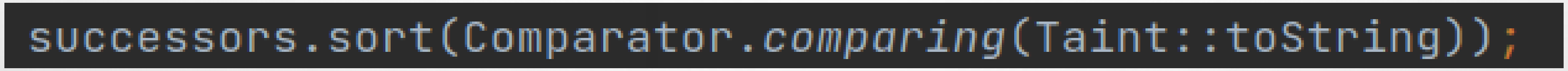
```
    @Override
    protected void flowThrough(Set<Taint> in, Unit unit, Set<Taint> out) {
```

- To save memory, I use some HashMaps to maintain a cache for UniqueStmts that have been created. And It requires some time for caching.



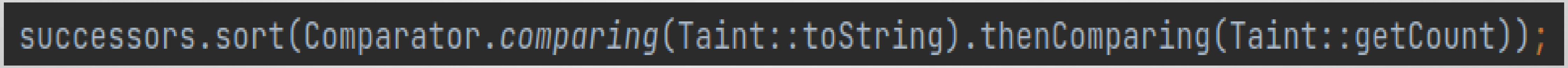
Solution

- Then I only need to add the info of statement count for comparison. That is, I change the comparison from



```
successors.sort(Comparator.comparing(Taint::toString));
```

to



```
successors.sort(Comparator.comparing(Taint::toString).thenComparing(Taint::getCount));
```

Content

- Problem
- Reason
- Solution
- Result
- Next step

Result

- I test the revised cFlow on hadoop_common 3.3.0 for 10 times and check the outputs.
- All output files contain 28746 lines and they only differ in execution time.

deterministic

- Both versions have 29 iterations to build taint propagation graph. However, the original one requires 50s, but the revised one requires 3min 20s.

Sacrifices performance

Content

- Problem
- Reason
- Solution
- Result
- Next step

Next step

- Test cFlow on more applications
- Consider implicit taint propagation
- Consider Alias Analysis