L0

1. iload 1
2. iload 2
3. iload 3
4. iadd
5. if\_cmpge L6
6. iload 2
7. iload 1
8. iload 3
9. iadd
10. if\_cmpge L6
11. iload 3
12. iload 1
13. iload 2
14. iadd
15. if\_cmpge L6

L1

1. iload 1
2. iload 2
3. if\_cmpne L3
4. iload 2
5. iload 3
6. if\_cmpne L3

L2

1. iconst\_1
2. ireturn

L3

1. iload 1
2. iload 2
3. if\_cmpeq L5
4. iload 1
5. iload 3
6. if\_cmpeq L5
7. iload 2
8. iload 3
9. if\_cmpeq L5

L4

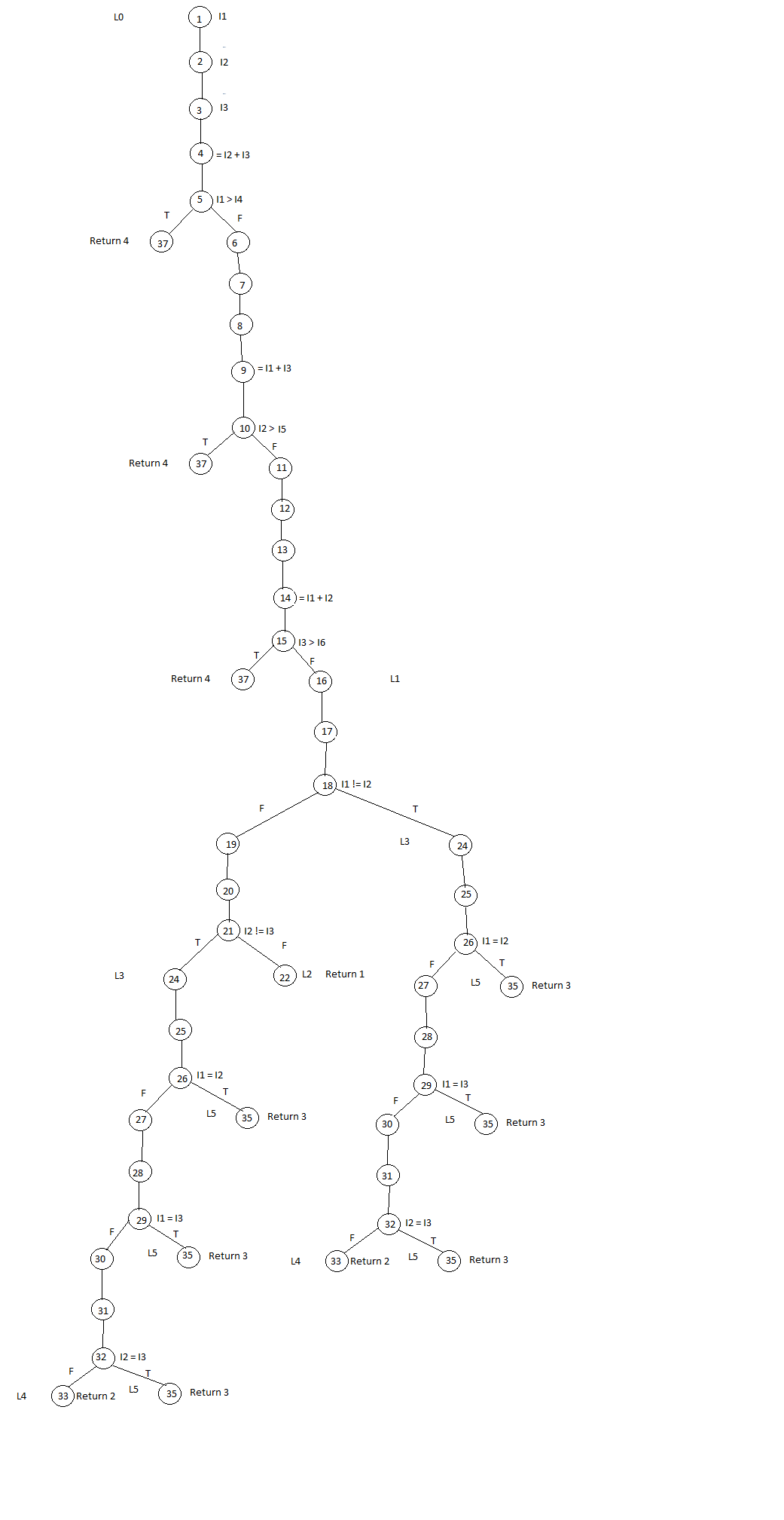
1. iconst\_2
2. ireturn

L5

1. iconst\_3
2. ireturn

L6

1. iconst\_4
2. ireturn



**Paths:**

|  |  |  |
| --- | --- | --- |
| **Triangle** | **Path ID** | **Path** |
| **Equilateral** | **1** |  |
| **Isosceles** | **2** |  |
|  | **3** |  |
|  | **4** |  |
|  | **5** |  |
|  | **6** |  |
|  | **7** |  |
| **Scalene** | **8** |  |
|  | **9** |  |
| **Not a Triangle** | **10** |  |
|  | **11** |  |
|  | **12** |  |

|  |  |  |
| --- | --- | --- |
| Path ID | Instructions | Dependencies |
| 1 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3 4. iload 2 5. iload 3 6. if\_cmpne L3   L2   1. iconst\_1 2. ireturn |  |
| 2 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5   L5   1. iconst\_3 2. ireturn |  |
| 3 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5 4. iload 1 5. iload 3 6. if\_cmpeq L5   L5   1. iconst\_3 2. ireturn |  |
| 4 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5 4. iload 1 5. iload 3 6. if\_cmpeq L5 7. iload 2 8. iload 3 9. if\_cmpeq L5   L5   1. iconst\_3 2. ireturn |  |
| 5 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3 4. iload 2 5. iload 3 6. if\_cmpne L3   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5   L5   1. iconst\_3 2. ireturn |  |
| 6 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3 4. iload 2 5. iload 3 6. if\_cmpne L3   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5 4. iload 1 5. iload 3 6. if\_cmpeq L5   L5   1. iconst\_3 2. ireturn |  |
| 7 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3 4. iload 2 5. iload 3 6. if\_cmpne L3   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5 4. iload 1 5. iload 3 6. if\_cmpeq L5 7. iload 2 8. iload 3 9. if\_cmpeq L5   L5   1. iconst\_3 2. ireturn |  |
| 8 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5 4. iload 1 5. iload 3 6. if\_cmpeq L5 7. iload 2 8. iload 3 9. if\_cmpeq L5   L4   1. iconst\_2 2. ireturn |  |
| 9 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L1   1. iload 1 2. iload 2 3. if\_cmpne L3 4. iload 2 5. iload 3 6. if\_cmpne L3   L2   1. iconst\_1 2. ireturn   L3   1. iload 1 2. iload 2 3. if\_cmpeq L5 4. iload 1 5. iload 3 6. if\_cmpeq L5 7. iload 2 8. iload 3 9. if\_cmpeq L5   L4   1. iconst\_2 2. ireturn |  |
| 10 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6   L6   1. iconst\_4 2. ireturn |  |
| 11 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6   L6   1. iconst\_4 2. ireturn |  |
| 12 | L0   1. iload 1 2. iload 2 3. iload 3 4. iadd 5. if\_cmpge L6 6. iload 2 7. iload 1 8. iload 3 9. iadd 10. if\_cmpge L6 11. iload 3 12. iload 1 13. iload 2 14. iadd 15. if\_cmpge L6   L6   1. iconst\_4 2. ireturn |  |

Code Implementation:

Com.gannon.executor.opcodes: all java file will implement Analyzable Interface and force implement dependencyAnalysing Method.

Create an DependencyDataHolder object. It should have ArrayListOfTree, tempVariableStack, tempVariableCounter, nodeCounter.

In the implementation of dependencyAnalysing method, make a use of tempVariableStack to hold the tempVariable Data same as JVM Stack does to hold the variables value. Whenever JVM Stack pops out the values during execution, we have to pops out the tempVariables from the stack and create a new tempVariable, which will be the root of the tree.

Pseudocode

Init two array lists a\_r and a\_v

Init two stack s\_r and s\_v

//a\_r: store relations

//a\_v: store temp computing values

For (each instruction)

if push (load) value in a\_r to s\_r

push corresponding variable name in a\_v to s\_r

end if

…

If pop (store) value in s\_v to a\_v

If(!exist)

Pop corresponding variable name r in s\_r to a\_r

Else

Move the r to correct position

End if

End for