See the Assessment Guide for information on how to interpret this report.

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     FAILED (1 warning)
PMD:          PASSED
Checkstyle:   PASSED

Correctness:  36/36 tests passed
Memory:       4/4 tests passed
Timing:       33/27 tests passed

Aggregate score: 104.44%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 60%, Timing: 10%, Memory: 20% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
1.2K Nov 18 09:52 Outcast.java
8.5K Nov 18 09:52 SAP.java
4.1K Nov 18 09:52 WordNet.java


********************************************************************************
*  COMPILING
********************************************************************************


% javac SAP.java
*-----------------------------------------------------------

% javac WordNet.java
*-----------------------------------------------------------

% javac Outcast.java
*-----------------------------------------------------------


================================================================


Checking the APIs of your programs.
*-----------------------------------------------------------
SAP:

WordNet:

Outcast:

================================================================


********************************************************************************
*  CHECKING STYLE AND COMMON BUG PATTERNS
********************************************************************************


% spotbugs *.class
*-----------------------------------------------------------
M C SUI_CONTAINS_BEFORE_ADD SUI: Method SAP.bfs(Queue, Map, Map, Set, Set) checks for an item in a set with contains, before using add()  At SAP.java:[line 258]
SpotBugs ends with 1 warning.


================================================================


% pmd .
*-----------------------------------------------------------


================================================================


% checkstyle *.java
*-----------------------------------------------------------

% custom checkstyle checks for SAP.java
*-----------------------------------------------------------

% custom checkstyle checks for WordNet.java
*-----------------------------------------------------------

% custom checkstyle checks for Outcast.java
*-----------------------------------------------------------


================================================================


********************************************************************************
*  TESTING CORRECTNESS
********************************************************************************
```

```
Testing correctness of SAP
*-----------------------------------------------------------
Running 20 total tests.

Test 1: check length() and ancestor() on fixed digraphs
  * digraph1.txt
  * digraph2.txt
  * digraph3.txt
  * digraph4.txt
  * digraph5.txt
  * digraph6.txt
  * digraph9.txt
==> passed

Test 2: check length() and ancestor() on WordNet digraph
  * 100 random vertex pairs in digraph-wordnet.txt
==> passed

Test 3: check length() and ancestor() on directed paths
  * 5
  * 10
  * 20
  * 50
  * 100
==> passed

Test 4: check length() and ancestor() on directed cycles
  * 5
  * 10
  * 20
  * 50
  * 100
==> passed

Test 5: check length() and ancestor() on complete graphs
  * 5
  * 10
  * 20
  * 50
==> passed

Test 6: check length() and ancestor() on tournament digraphs
  * 5
  * 10
  * 20
  * 50
==> passed

Test 7: check length() and ancestor() on complete binary trees
  * 5
  * 10
  * 20
  * 50
  * 100
==> passed

Test 8: check length() and ancestor() on random DAGs
  * 5 vertices, 8 edges
  * 10 vertices, 40 edges
  * 20 vertices, 100 edges
==> passed

Test 9: check length() and ancestor() on random rooted-in DAGs
  * 5 vertices, 8 edges
  * 10 vertices, 40 edges
  * 20 vertices, 100 edges
==> passed

Test 10: check length() and ancestor() on random rooted-out DAGs
  * 5 vertices, 8 edges
  * 10 vertices, 40 edges
  * 20 vertices, 100 edges
==> passed

Test 11: check length() and ancestor() on random rooted-in trees
  * 5 vertices
  * 10 vertices
  * 20 vertices
==> passed

Test 12: check length() and ancestor() on random rooted-out trees
  * 5 vertices
  * 10 vertices
  * 20 vertices
==> passed

Test 13: check length() and ancestor() on random simple digraphs
  * 5 vertices, 8 edges
  * 10 vertices, 40 edges
  * 20 vertices, 100 edges
==> passed

Test 14: check whether two SAP objects can be created at the same time
  * digraph1.txt and digraph2.txt
  * digraph3.txt and digraph4.txt
  * digraph5.txt and digraph6.txt
  * digraph2.txt and digraph1.txt
==> passed

Test 15: check whether SAP is immutable
  * digraph1.txt
  * digraph2.txt
  * digraph3.txt
  * digraph4.txt
  * digraph5.txt
  * digraph6.txt
  * digraph-ambiguous-ancestor.txt
==> passed
```

```
Test 16: check length() and ancestor() with iterable arguments
  * 100 random subsets of 1 and 1 vertices in digraph-wordnet.txt
  * 100 random subsets of 1 and 2 vertices in digraph-wordnet.txt
  * 100 random subsets of 2 and 1 vertices in digraph-wordnet.txt
  * 100 random subsets of 2 and 2 vertices in digraph-wordnet.txt
  * 100 random subsets of 3 and 11 vertices in digraph-wordnet.txt
  * 100 random subsets of 11 and 3 vertices in digraph-wordnet.txt
==> passed

Test 17: check length() and ancestor() with zero-length iterable arguments
  * 100 random subsets of 0 and 5 vertices in digraph-wordnet.txt
  * 100 random subsets of 5 and 0 vertices in digraph-wordnet.txt
  * 100 random subsets of 0 and 0 vertices in digraph-wordnet.txt
==> passed

Test 18: check length() and ancestor() with invalid arguments
  * G = digraph1.txt v = -1, w = 0
  * G = digraph1.txt v = 0, w = -1
  * G = digraph1.txt v = 13, w = 0
  * G = digraph1.txt v = 0, w = 13
==> passed

Test 19: check iterable versions of length() and ancestor() with invalid arguments
  * G = digraph1.txt, v = { 0, 7, 9, 12 }, w = null
  * G = digraph1.txt, v = null, w = { 1, 2, 4, 5, 10 }
  * G = digraph1.txt, v = null, w = null
  * G = digraph1.txt, v = { 0, 7, 9, 12, -1 }, w = { 1, 2, 4, 5, 10 }
  * G = digraph1.txt, v = { 0, 7, 9, 12 }, w = { 1, 2, -1, 4, 5, 10 }
  * G = digraph1.txt, v = { 13, 0, 7, 9, 12 }, w = { 1, 2, 4, 5, 10 }
  * G = digraph1.txt, v = { 0, 7, 9, 12 }, w = { 1, 2, 4, 5, 13, 10 }
  * G = digraph1.txt, v = { 0, null, 7, 9, 12 }, w = { 1, 2, 4, 5, 10 }
  * G = digraph1.txt, v = { 0, 7, 9, 12 }, w = { 1, 2, 4, null, 5, 10 }
==> passed

Test 20: random calls to both version of length() and ancestor(),
         with probabilities p1 and p2, respectively
  * random calls in a random rooted DAG (20 vertices, 100 edges)
    (p1 = 0.5, p2 = 0.5)
  * random calls in a random digraph (20 vertices, 100 edges)
    (p1 = 0.5, p2 = 0.5)
==> passed


Total: 20/20 tests passed!


================================================================
********************************************************************************
*  TESTING CORRECTNESS (substituting reference SAP)
********************************************************************************

Testing correctness of WordNet
*-----------------------------------------------------------
Running 14 total tests.

Test 1: check distance() with random noun pairs
  * 1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.txt
==> passed

Test 2: check distance() with all noun pairs
  * synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  * synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  * synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  * synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAncestor.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  * synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAncestor.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.txt
==> passed

Test 3: check distance() with random noun pairs
  * 1000 pairs using synsets = synsets100-subgraph.txt; hypernyms = hypernyms100-subgraph.txt
  * 1000 pairs using synsets = synsets500-subgraph.txt; hypernyms = hypernyms500-subgraph.txt
  * 1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms = hypernyms1000-subgraph.txt
==> passed

Test 4: check sap() with random noun pairs
  * 1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.txt
==> passed

Test 5: check sap() with all noun pairs
  * synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  * synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  * synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  * synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAncestor.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  * synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAncestor.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.txt
==> passed

Test 6: check sap() with random noun pairs
  * 1000 pairs using synsets = synsets100-subgraph.txt; hypernyms = hypernyms100-subgraph.txt
  * 1000 pairs using synsets = synsets500-subgraph.txt; hypernyms = hypernyms500-subgraph.txt
  * 1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms = hypernyms1000-subgraph.txt
==> passed

Test 7: check whether WordNet is immutable
  * synsets = synsets.txt; hypernyms = hypernyms.txt
==> passed

Test 8: check constructor when input is not a rooted DAG
  * synsets3.txt, hypernyms3InvalidTwoRoots.txt
  * synsets3.txt, hypernyms3InvalidCycle.txt
  * synsets6.txt, hypernyms6InvalidTwoRoots.txt
  * synsets6.txt, hypernyms6InvalidCycle.txt
  * synsets6.txt, hypernyms6InvalidCycle+Path.txt
```

```
==> passed

Test 9: check isNoun()
  * synsets = synsets.txt; hypernyms = hypernyms.txt
  * synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 10: check nouns()
  * synsets = synsets.txt; hypernyms = hypernyms.txt
  * synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  * synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 11: check whether two WordNet objects can be created at the same time
  * synsets1 = synsets15.txt; hypernyms1 = hypernyms15Tree.txt
    synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
  * synsets1 = synsets.txt; hypernyms1 = hypernyms.txt
    synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
==> passed

Test 12: call distance() and sap() with invalid arguments
  * synsets15.txt, hypernyms15Tree.txt, nounA = "x", nounB = "b"
  * synsets15.txt, hypernyms15Tree.txt, nounA = "b", nounB = "x"
  * synsets15.txt, hypernyms15Tree.txt, nounA = "x", nounB = "a"
  * synsets15.txt, hypernyms15Tree.txt, nounA = "x", nounB = "x"
  * synsets15.txt, hypernyms15Tree.txt, nounA = "a", nounB = null
  * synsets15.txt, hypernyms15Tree.txt, nounA = null, nounB = "a"
  * synsets15.txt, hypernyms15Tree.txt, nounA = null, nounB = null
  * synsets15.txt, hypernyms15Tree.txt, nounA = "x", nounB = null
  * synsets15.txt, hypernyms15Tree.txt, nounA = null, nounB = "x"
==> passed

Test 13: call isNoun() with a null argument
  * synsets15.txt, hypernyms15Path.txt
==> passed

Test 14: random calls to isNoun(), distance(), and sap(), with
         probabilities p1, p2, and p3, respectively
  * 100 random calls (p1 = 0.5, p2 = 0.5, p3 = 0.0)
  * 100 random calls (p1 = 0.5, p2 = 0.0, p3 = 0.5)
  * 100 random calls (p1 = 0.0, p2 = 0.5, p3 = 0.5)
  * 100 random calls (p1 = 0.2, p2 = 0.4, p3 = 0.4)
==> passed


Total: 14/14 tests passed!


================================================================
********************************************************************************
*  TESTING CORRECTNESS (substituting reference SAP and WordNet)
********************************************************************************

Testing correctness of Outcast
*-----------------------------------------------------------
Running 2 total tests.

Test 1: check outcast() on WordNet digraph
        (synsets.txt and hypernyms.txt)
  * outcast2.txt
  * outcast3.txt
  * outcast4.txt
  * outcast5.txt
  * outcast5a.txt
  * outcast7.txt
  * outcast8.txt
  * outcast8a.txt
  * outcast8b.txt
  * outcast8c.txt
  * outcast9.txt
  * outcast9a.txt
  * outcast10.txt
  * outcast10a.txt
  * outcast11.txt
  * outcast12.txt
  * outcast12a.txt
  * outcast17.txt
  * outcast20.txt
  * outcast29.txt
==> passed

Test 2: check outcast() on WordNet subgraph
        (synsets50000-subgraph.txt and hypernyms50000-subgraph.txt)
  * outcast2.txt
  * outcast3.txt
  * outcast5.txt
  * outcast5a.txt
  * outcast7.txt
  * outcast8.txt
  * outcast8b.txt
  * outcast8c.txt
  * outcast9.txt
  * outcast10.txt
  * outcast11.txt
==> passed


Total: 2/2 tests passed!


================================================================
********************************************************************************
*  MEMORY
********************************************************************************

Analyzing memory of SAP
*-----------------------------------------------------------
```

```
Running 1 total tests.

digraph G             = digraph-wordnet.txt
vertices in G         = 82192
edges    in G         = 84505
student     memory    = 8348176 bytes
reference   memory    = 10320712 bytes
ratio                 = 0.81
maximum allowed ratio = 2.50

Total: 1/1 tests passed!


================================================================



Analyzing memory of WordNet
*-----------------------------------------------------------
Running 3 total tests.

Test 1a: check memory of WordNet object
  * synsets = synsets1000-subgraph.txt; hypernyms = hypernyms1000-subgraph.txt
    - number of vertices in digraph = 1000
    - number of edges    in digraph = 1008
    - student   memory              = 732488 bytes
    - reference memory              = 1441648 bytes
    - student / reference ratio     = 0.5
    - maximum allowed rato          = 2.0

==> passed

Test 1b: check memory of WordNet object
  * synsets = synsets5000-subgraph.txt; hypernyms = hypernyms5000-subgraph.txt
    - number of vertices in digraph = 5000
    - number of edges    in digraph = 5059
    - student   memory              = 3570008 bytes
    - reference memory              = 7042192 bytes
    - student / reference ratio     = 0.5
    - maximum allowed rato          = 2.0

==> passed

Test 1c: check memory of WordNet object
  * synsets = synsets10000-subgraph.txt; hypernyms = hypernyms10000-subgraph.txt
    - number of vertices in digraph = 10000
    - number of edges    in digraph = 10087
    - student   memory              = 8412104 bytes
    - reference memory              = 16174760 bytes
    - student / reference ratio     = 0.5
    - maximum allowed rato          = 2.0

==> passed

Total: 3/3 tests passed!


================================================================



********************************************************************************
*  TIMING
********************************************************************************

Timing SAP
*-----------------------------------------------------------
Running 14 total tests.

Test 1: time SAP constructor
   *  digraph-wordnet.txt
      -  student solution time =  0.01 seconds
      -  maximum allowed  time =  1.00 seconds
==> passed

Test 2a-c: time length() and ancestor() with random pairs of vertices
  *  digraph-wordnet.txt
     -  reference solution calls per second:  667653.00
     -  student   solution calls per second:  392648.00
     -  reference / student ratio:                 1.70

=> passed      student <= 50000x reference
=> passed      student <= 10000x reference
=> passed      student <=  5000x reference
=> passed      student <=  1000x reference
=> BONUS       student <=   100x reference
=> BONUS       student <=    10x reference
=> BONUS       student <=     2x reference

Test 3a-c: time length() and ancestor() with random subsets of 5 vertices
  *  digraph-wordnet.txt
     -  reference solution calls per second:  188313.00
     -  student   solution calls per second:  102618.00
     -  reference / student ratio:                 1.84

=> passed      student <= 10000x reference
=> passed      student <=  5000x reference
=> passed      student <=  1000x reference
=> passed      student <=   500x reference
=> BONUS       student <=    10x reference
=> BONUS       student <=     2x reference

Test 4a-c: time length() and ancestor() with random subsets of 100 vertices
  *  digraph-wordnet.txt
     -  reference solution calls per second:   10793.00
     -  student   solution calls per second:    6038.00
     -  reference / student ratio:                 1.79
```

```
=> passed      student <= 10000x reference
=> passed      student <=  5000x reference
=> passed      student <=  1000x reference
=> passed      student <=   500x reference
=> BONUS       student <=     2x reference

Test 5: Time 10 calls to length() and ancestor() on random path graphs
        (must handle V = 65536 in under 2 seconds)

             V  seconds
        ---------------
         32768     0.16
         65536     0.50
==> passed


Total: 20/14 tests passed!


================================================================



********************************************************************************
*  TIMING (substituting reference SAP)
********************************************************************************

Timing WordNet
*-----------------------------------------------------------
Running 11 total tests.

Test 1: check that exactly two In object created
        (one for synsets file and one for hypernyms file)
==> passed

Test 2: count number of SAP operations when constructing a WordNet object
        and calling distance() and sap() three times each
  * calls to constructor = 1
  * calls to length()    = 3
  * calls to ancestor()  = 3

==> passed

Test 3: count Digraph operations during WordNet constructor
  * synsets = synsets.txt; hypernyms = hypernyms.txt
  * number of synsets    = 82192
  * number of hypernyms  = 84505
  * calls to constructor = 2
  * calls to addEdge()   = 84505
  * calls to adj()       = 82192
  * calls to outdegree() = 82192
  * calls to indegree()  = 82192
  * calls to reverse()   = 0
  * calls to toString()  = 0

==> passed

Test 4: count Digraph operations during 1000 calls each
        to distance() and sap()
  * synsets = synsets.txt; hypernyms = hypernyms.txt
  * calls to constructor = 0
  * calls to addEdge()   = 0
  * calls to adj()       = 46078
  * calls to reverse()   = 0
  * calls to toString()  = 0

==> passed

Test 5: time WordNet constructor
  * synsets = synsets.txt; hypernyms = hypernyms.txt
    - student constructor time =  0.27 seconds
    - maximum allowed     time = 10.00 seconds

==> passed

Test 6a-e: time sap() and distance() with random nouns
  * synsets = synsets.txt; hypernyms = hypernyms.txt
    - reference solution calls per second:  187226.75
    - student   solution calls per second:  216050.00
    - reference / student ratio:                 0.87

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> passed    student <=   100x reference
=> passed    student <=    10x reference
=> passed    student <=     5x reference

Test 7: time isNoun() with random nouns
  * synsets = synsets.txt; hypernyms = hypernyms.txt
    - reference solution calls per second:  878597.00
    - student   solution calls per second:  708843.00
    - reference / student ratio:                 1.24
    - allowed ratio:                             4.00
==> passed

Total: 11/11 tests passed!


================================================================



********************************************************************************
*  TIMING (substituting reference SAP and WordNet)
********************************************************************************

Timing Outcast
*-----------------------------------------------------------
```

```
Running 2 total tests.

Test 1: count calls to methods in WordNet
 * outcast4.txt
 * outcast10.txt
 * outcast29.txt
==> passed

Test 2: timing calls to outcast() for various outcast files

Total time must not exceed 1.0 seconds.

    filename       n     time
-----------------------------
   outcast4.txt    4     0.00
   outcast5.txt    5     0.00
  outcast5a.txt    5     0.00
   outcast5.txt    5     0.00
   outcast7.txt    7     0.00
   outcast8.txt    8     0.00
  outcast8a.txt    8     0.00
  outcast8b.txt    8     0.00
  outcast8c.txt    8     0.00
   outcast9.txt    9     0.00
  outcast9a.txt    9     0.00
  outcast10.txt   10     0.00
 outcast10a.txt   10     0.00
  outcast11.txt   11     0.00
  outcast12.txt   12     0.00
 outcast12a.txt   12     0.00
  outcast20.txt   20     0.00
  outcast29.txt   29     0.01

Total elapsed time: 0.01 seconds

==> passed


Total: 2/2 tests passed!


================================================================
```