README

Junwen Bu

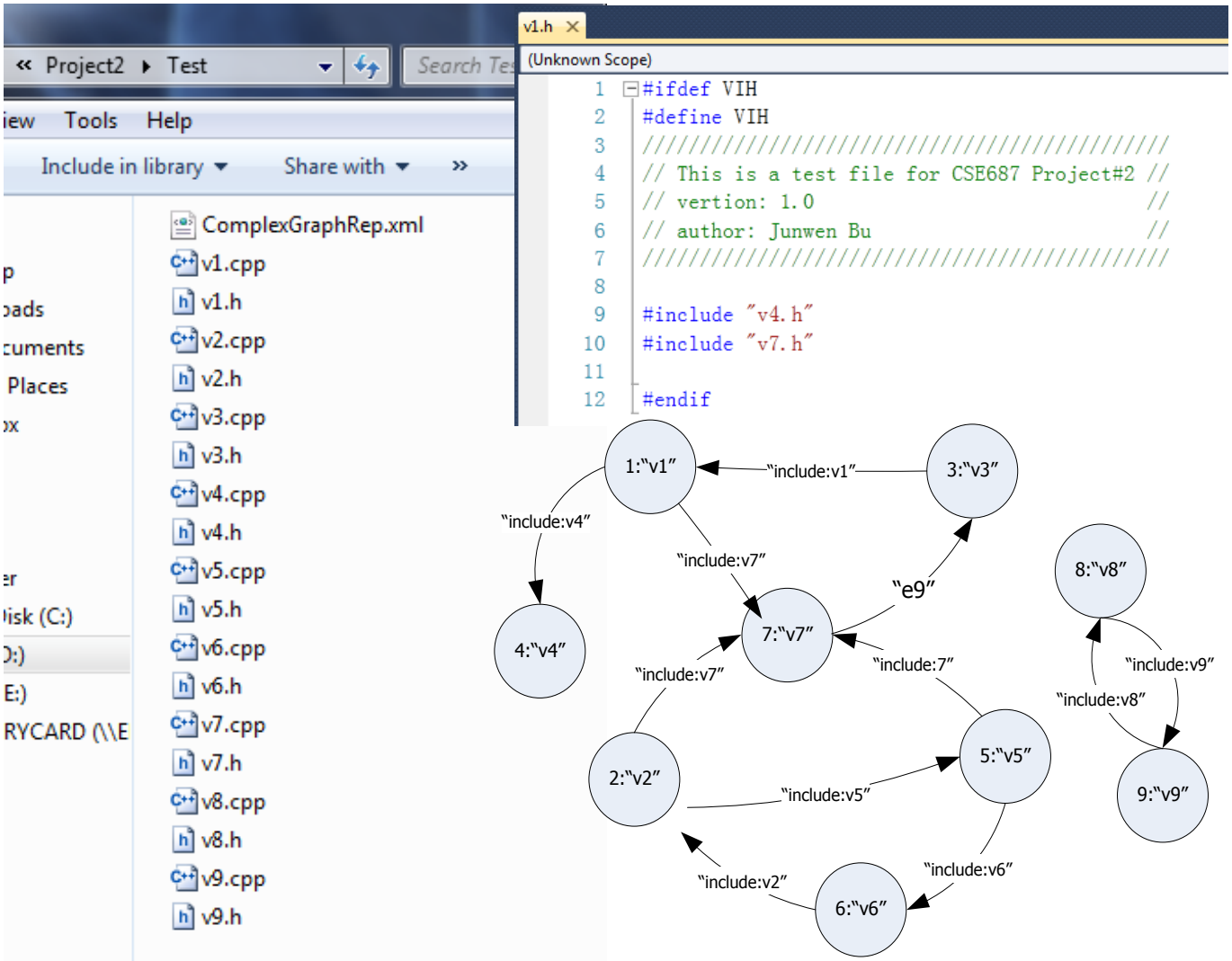
Run File Dependency Test

Test File Description

compile.bat and **run.bat** are in directory: **./Project2/Project2**

In ./Project2/**Test** Folder there are several test **.cpp** and **.h** test files:

In the graph figure, **#:"v#"** means **[vertex id]: [value of vertex]** and **"include:v#"** is **[edge value].**



Run Test

1. run **compile.bat**
2. run **run.bat**

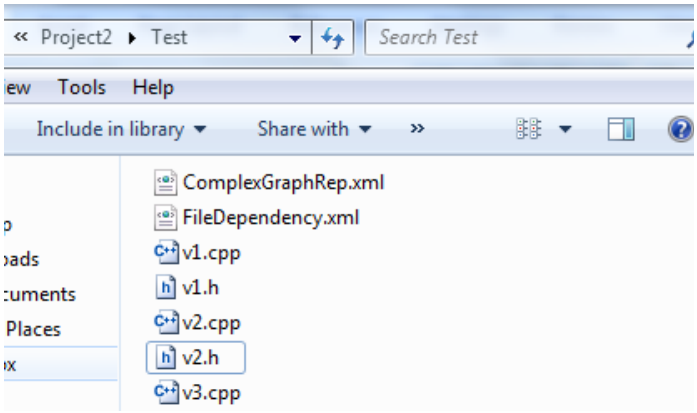
:run.bat

..\Debug\Project2.exe ../Test FileDependency.xml

In run.bat **../Test** is the path where your test .h and .cpp files in and the program will analyze those files then generate **FileDependency.xml** and write file dependency relationship into **FileDependency.xml**.

After doing these things, the program will **create a graph by reading FileDependency.xml** and do operations on the graph. (*The picture shows that after running **run.bat**, **FileDependency.xml** is generated in the **Test** folder.*)

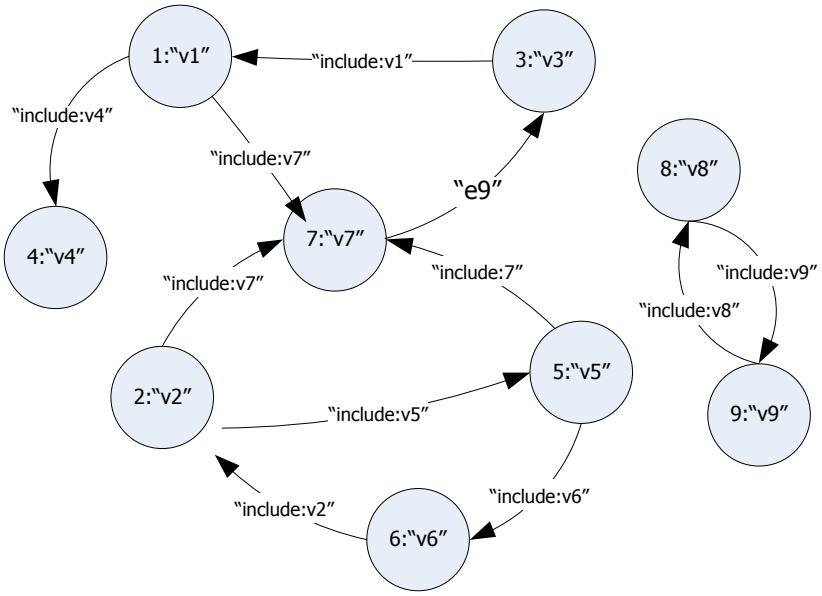
```
<?xml version="1.0"?>
- <graph>
        <!-- Each vertex and edge have an id used as the target for same searches. -->
        <!-- The vertex bodies are composed of edge elements. -->
        <!-- Reprecent files as a graph for CSE687 - OOD, Project #2 -->
    <name>FileDependency</name>
  - <vertex value="v1" id="1">
        <edge value="include:v4" id="1">4</edge>
        <edge value="include:v7" id="2">7</edge>
    </vertex>
  - <vertex value="v2" id="2">
        <edge value="include:v5" id="3">5</edge>
        <edge value="include:v7" id="4">7</edge>
    </vertex>
  - <vertex value="v3" id="3">
        <edge value="include:v1" id="5">1</edge>
    </vertex>
    <vertex value="v4" id="4"/>
  - <vertex value="v5" id="5">
        <edge value="include:v6" id="6">6</edge>
        <edge value="include:v7" id="7">7</edge>
    </vertex>
  - <vertex value="v6" id="6">
        <edge value="include:v2" id="8">2</edge>
    </vertex>
  - <vertex value="v7" id="7">
        <edge value="include:v3" id="9">3</edge>
    </vertex>
  - <vertex value="v8" id="8">
        <edge value="include:v9" id="10">9</edge>
    </vertex>
  - <vertex value="v9" id="9">
        <edge value="include:v8" id="11">8</edge>
    </vertex>
</graph>
```



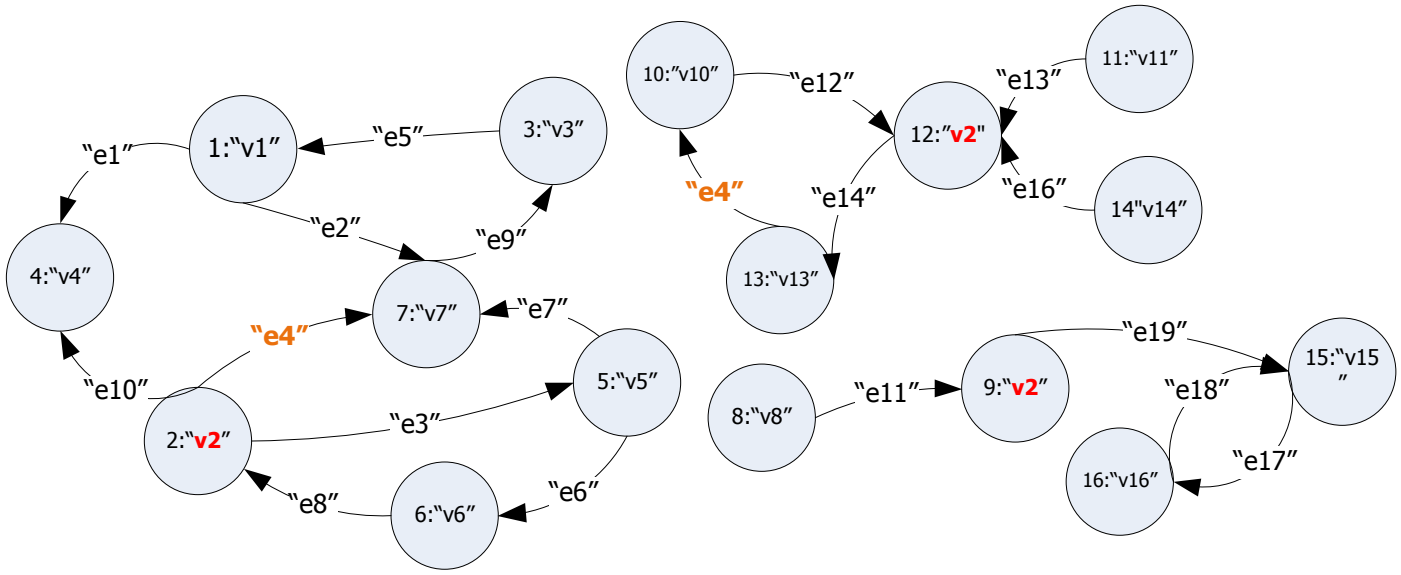
Test Result

The results of running algorithms on the graph generated from the FileDependency.xml:

```
========= < Start Graph Search > ========
=========================================
Do Depth-First Search in directed Graphs:
Vertex id sequence:
1 4 7 3 2 5 6 8 9

========= < Strong Components > =========
=========================================
Vertex id Set1 :  4
Vertex id Set2 :  3 7 1
Vertex id Set3 :  6 5 2
Vertex id Set4 :  9 8

========= < Result of Condensation > ====
=========================================
Display condensed adjacency list:
-----------------------------------------
Id: 4 condensed from vertex:  4
Id: 1 condensed from vertex:  3 7 1
Id: 2 condensed from vertex:  6 5 2
Id: 8 condensed from vertex:  9 8

Display condensed edge list:
-----------------------------------------
edge from 1 to 4 :  original edge <1,4> value: include:v4
edge from 2 to 1 :  original edge <2,7> value: include:v7

==<Topological Order on Condensed Graph>=
=========================================
8 2 1 4

======== < Partitions of Vertex id > ====
=========================================
Partition id Set1:  1 2 3 4 5 6 7
Partition id Set2:  8 9

========= < Test Global Function > =======
getVertexIdByValue -- Find vertex value: "v2"
 vertexID=2;
getEdgeInfoByValue -- Find edge value: "include:v4"
 edge: <1,4>;
```



Run Complex Graph Test



run **testComplexGraph.bat** in directory: **./Project2/Project2**

Graph XML File: **ComplexGraphRep.xml**

```
========= < Start Graph Search > ========
=========================================
Do Depth-First Search in directed Graphs:
Vertex id sequence:
1 4 7 3 2 5 6 8 9 15 16 10 12 13 11 14

========= < Strong Components > =========
=========================================
Vertex id Set1 :  4
Vertex id Set2 :  3 7 1
Vertex id Set3 :  6 5 2
Vertex id Set4 :  16 15
Vertex id Set5 :  9
Vertex id Set6 :  8
Vertex id Set7 :  13 12 10
Vertex id Set8 :  11
Vertex id Set9 :  14

========= < Result of Condensation > ====
=========================================
Display condensed adjacency list:
-----------------------------------------
Id: 4 condensed from vertex:  4
Id: 1 condensed from vertex:  3 7 1
Id: 2 condensed from vertex:  6 5 2
Id: 15 condensed from vertex:  16 15
Id: 9 condensed from vertex:  9
Id: 8 condensed from vertex:  8
Id: 10 condensed from vertex:  13 12 10
Id: 11 condensed from vertex:  11
Id: 14 condensed from vertex:  14
```

```
Display condensed edge list:
-----------------------------------------
edge from 1 to 4 :  original edge <1,4) value: e1
edge from 2 to 1 :  original edge <2,7) value: e4
edge from 2 to 4 :  original edge <2,4) value: e10
edge from 8 to 9 :  original edge <8,9) value: e11
edge from 9 to 15 :  original edge <9,15) value: e19
edge from 11 to 10 :  original edge <11,12) value: e13
edge from 14 to 10 :  original edge <14,12) value: e16

==<Topological Order on Condensed Graph>=
=========================================
14 11 10 8 9 15 2 1 4

======== < Partitions of Vertex id > ====
=========================================
Partition id Set1:  1 2 3 4 5 6 7
Partition id Set2:  8 9 15 16
Partition id Set3:  10 11 12 13 14

========= < Test Global Function > =======
getVertexIdByValue -- Find vertex value: "v2"
 vertexID=2;  vertexID=9;  vertexID=12;
getEdgeInfoByValue -- Find edge value: "include:v4"
 edge: <2,7);  edge: <13,10);
```