1 Ordre topologique

```
void dfsVisit(graphNode node) {
   graphNode k = null;

   node.color = GREY_COLOR;
   node.dtime = time++;

   System.out.println("DFS DISCOVERY: " + node.nodeId);
   System.out.println("DTIME: " + node.dtime);

   node.kids.first();
   while (node.kids.currentIsValid()) {
      k = node.kids.getCurrent();
      if (k.color == WHITE_COLOR) {
         k.pred = node;
         System.out.println("KID: " + k.nodeId +
                                 " PRED: " + node.nodeId);
         dfsVisit(k);
      }
      node.kids.next();
   }
   node.color = BLACK_COLOR;
   node.ftime = time++;
   System.out.println("DFS CLOSING: " + node.nodeId);
   System.out.println("FTIME: " + node.ftime);
   topolOrder.addLast(node);
};
```

```
void computeTopolOrder(graph g) {

   int i = UNDEF_VAL;
   graphNode node = null;

   g.first();
   while (g.currentIsValid()) {
      node = g.getCurrent();

      node.color = WHITE_COLOR;
      node.dtime = 0;
      node.ftime = 0;
      node.pred = null;

      g.next();
   }

   time = 0;
   g.first();
   while (g.currentIsValid()) {
      node = g.getCurrent();

      if (node.color == WHITE_COLOR)
         dfsVisit(node);

      g.next();
   }
}
```

1.1 Exemple

```
NODE: 0
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 1
K: 3
PARENTS:
P: 2

NODE: 1
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 3
K: 4
K: 5
PARENTS:
P: 0

NODE: 2
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 0
K: 4
K: 6
PARENTS:

NODE: 3
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 4
PARENTS:
P: 0
P: 1

NODE: 4
COLOR: -9999
```

```
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
PARENTS:
P: 1
P: 2
P: 3

NODE: 5
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 6
PARENTS:
P: 1

NODE: 6
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
PARENTS:
P: 2
P: 5

DFS DISCOVERY: 0
DTIME: 0
KID: 1 PRED: 0
DFS DISCOVERY: 1
DTIME: 1
KID: 3 PRED: 1
DFS DISCOVERY: 3
DTIME: 2
KID: 4 PRED: 3
DFS DISCOVERY: 4
DTIME: 3
DFS CLOSING: 4
FTIME: 4
DFS CLOSING: 3
FTIME: 5
KID: 5 PRED: 1
DFS DISCOVERY: 5
DTIME: 6
KID: 6 PRED: 5
DFS DISCOVERY: 6
DTIME: 7
DFS CLOSING: 6
```

```
FTIME: 8
DFS CLOSING: 5
FTIME: 9
DFS CLOSING: 1
FTIME: 10
DFS CLOSING: 0
FTIME: 11
DFS DISCOVERY: 2
DTIME: 12
DFS CLOSING: 2
FTIME: 13

TOPOLOGICAL ORDER:
POS: 1 NODE: 4
POS: 2 NODE: 3
POS: 3 NODE: 6
POS: 4 NODE: 5
POS: 5 NODE: 1
POS: 6 NODE: 0
POS: 7 NODE: 2
```

2 Composantes fortement connexes - SCC

```
void dfsVisit(int compId, graphNode node, int[] fNode) {
   graphNode k = null;

   node.componentId = compId;
   node.color = GREY_COLOR;

   System.out.println("DFS DISCOVERY: " + node.nodeId);
   System.out.println("DTIME: " + time);
   System.out.println("COMP: " + node.componentId);

   node.kids.first();
   while (node.kids.currentIsValid()) {
   k = node.kids.getCurrent();
   if (k.color == WHITE_COLOR) {
         k.pred = node;
         System.out.println("KID: " + k.nodeId +
         " PRED: " + node.nodeId);
         dfsVisit(compId, k, fNode);
      }
      node.kids.next();
   }
   node.color = BLACK_COLOR;
   node.ftime = time++;
   fNode[node.nodeId] = node.ftime;
   System.out.println("DFS CLOSING: " + node.nodeId);
   System.out.println("FTIME: " + node.ftime);
};
```

```
void scc(graph g) {

   int i = UNDEF_VAL;
   graphNode node = null;
   int compId = UNDEF_VAL;
   int[] fNode = new int[g.size()];
   graphNode[] order = new graphNode[g.size()];

   g.first();
   while (g.currentIsValid()) {
      node = g.getCurrent();

      node.color = WHITE_COLOR;
      node.dtime = 0;
      node.ftime = 0;
      node.pred = null;

      g.next();
   }


   //
   // compute finishing times
   //
   time = 0;
   g.first();
   while (g.currentIsValid()) {
      node = g.getCurrent();

      if (node.color == WHITE_COLOR)
         dfsVisit(UNDEF_VAL, node, fNode);

      g.next();
   }

   System.out.println("FNODE:");
   for (i=0;i¡fNode.length;i++) {
      System.out.println("\tNODE: " + i +
      " POS: " + fNode[i]);
   }
   System.out.println();
```

```
//
// reverse the graph
//

gRev = g.reverse();
gRev.print();

//
// compute order for reversed graph
//

gRev.first();
while (gRev.currentIsValid()) {
   node = gRev.getCurrent();

   order[fNode[node.nodeId]] = node;

   gRev.next();
}

System.out.println("ORDER:");
for (i=0;i¡fNode.length;i++) {
   System.out.println("\tPOS: " + i +
   " NODE: " + order[i].nodeId);
}
System.out.println();
```

```
    //
    // initialize reversed graph
    //

    gRev.first();
    while (gRev.currentIsValid()) {
       node = gRev.getCurrent();

       node.color = WHITE_COLOR;
       node.dtime = 0;
       node.ftime = 0;
       node.pred = null;

       gRev.next();
    }

    //
    // DFS traverse reversed graph
    // starting from nodes with decreasing
    // finishing times
    //

    compId = 0;
    for (i = (order.length - 1); i ¿= 0; i–) {

       node = order[i];

       System.out.println(”REVERSED ORDERED NODE: ” + node.nodeId);

       if (node.color == WHITE_COLOR) {
          dfsVisit(compId, node, fNode);
          compId++;
       }
    }

    gRev.print();
 }
```

2.1 Exemple

```
NODE: 0
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 2
PARENTS:
P: 3

NODE: 1
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 8
PARENTS:
P: 5

NODE: 2
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 5
PARENTS:
P: 0
P: 5
P: 6

NODE: 3
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 0
PARENTS:
P: 4

NODE: 4
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 3
K: 11
PARENTS:
P: 7
```

```
NODE: 5
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 1
K: 2
PARENTS:
P: 2

NODE: 6
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 2
PARENTS:
P: 9

NODE: 7
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 4
PARENTS:
P: 10
P: 11

NODE: 8
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 9
PARENTS:
P: 1

NODE: 9
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 6
PARENTS:
P: 8
P: 12
```

```
NODE: 10
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 7
K: 12
PARENTS:

NODE: 11
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 7
PARENTS:
P: 4

NODE: 12
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 9
PARENTS:
P: 10
```

```
DFS DISCOVERY: 0
DTIME: 0
COMP: -9999
KID: 2 PRED: 0
DFS DISCOVERY: 2
DTIME: 0
COMP: -9999
KID: 5 PRED: 2
DFS DISCOVERY: 5
DTIME: 0
COMP: -9999
KID: 1 PRED: 5
DFS DISCOVERY: 1
DTIME: 0
COMP: -9999
KID: 8 PRED: 1
DFS DISCOVERY: 8
DTIME: 0
COMP: -9999
KID: 9 PRED: 8
DFS DISCOVERY: 9
DTIME: 0
COMP: -9999
KID: 6 PRED: 9
DFS DISCOVERY: 6
DTIME: 0
COMP: -9999
DFS CLOSING: 6
FTIME: 0
DFS CLOSING: 9
FTIME: 1
DFS CLOSING: 8
FTIME: 2
DFS CLOSING: 1
FTIME: 3
DFS CLOSING: 5
FTIME: 4
DFS CLOSING: 2
FTIME: 5
DFS CLOSING: 0
FTIME: 6
DFS DISCOVERY: 3
DTIME: 7
COMP: -9999
DFS CLOSING: 3
FTIME: 7
DFS DISCOVERY: 4
DTIME: 8
COMP: -9999
KID: 11 PRED: 4
DFS DISCOVERY: 11
DTIME: 8
COMP: -9999
KID: 7 PRED: 11
DFS DISCOVERY: 7
DTIME: 8
COMP: -9999
DFS CLOSING: 7
FTIME: 8
DFS CLOSING: 11
FTIME: 9
DFS CLOSING: 4
FTIME: 10
```

```
DFS DISCOVERY: 10
DTIME: 11
COMP: -9999
KID: 12 PRED: 10
DFS DISCOVERY: 12
DTIME: 11
COMP: -9999
DFS CLOSING: 12
FTIME: 11
DFS CLOSING: 10
FTIME: 12

FNODE:
NODE: 0 POS: 6
NODE: 1 POS: 3
NODE: 2 POS: 5
NODE: 3 POS: 7
NODE: 4 POS: 10
NODE: 5 POS: 4
NODE: 6 POS: 0
NODE: 7 POS: 8
NODE: 8 POS: 2
NODE: 9 POS: 1
NODE: 10 POS: 12
NODE: 11 POS: 9
NODE: 12 POS: 11

ORDER:
NODE: 6 POS: 0
NODE: 9 POS: 1
NODE: 8 POS: 2
NODE: 1 POS: 3
NODE: 5 POS: 4
NODE: 2 POS: 5
NODE: 0 POS: 6
NODE: 3 POS: 7
NODE: 7 POS: 8
NODE: 11 POS: 9
NODE: 4 POS: 10
NODE: 12 POS: 11
NODE: 10 POS: 12
```

```
NODE: 0
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 3
PARENTS:
P: 2

NODE: 1
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 5
PARENTS:
P: 8

NODE: 2
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 0
K: 5
K: 6
PARENTS:
P: 5

NODE: 3
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 4
PARENTS:
P: 0

NODE: 4
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 7
PARENTS:
P: 3
P: 11
```

```
NODE: 5
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 2
PARENTS:
P: 1
P: 2

NODE: 6
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 9
PARENTS:
P: 2

NODE: 7
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 10
K: 11
PARENTS:
P: 4

NODE: 8
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 1
PARENTS:
P: 9

NODE: 9
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 8
K: 12
PARENTS:
P: 6
```

```
NODE: 10
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
PARENTS:
P: 7
P: 12

NODE: 11
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 4
PARENTS:
P: 7

NODE: 12
COLOR: -9999
DTIME: -9999
FTIME: -9999
COMPONENT: -9999
PRED: UNDEFINED
KIDS:
K: 10
PARENTS:
P: 9
```

```
ORDER:
POS: 0 NODE: 6
POS: 1 NODE: 9
POS: 2 NODE: 8
POS: 3 NODE: 1
POS: 4 NODE: 5
POS: 5 NODE: 2
POS: 6 NODE: 0
POS: 7 NODE: 3
POS: 8 NODE: 7
POS: 9 NODE: 11
POS: 10 NODE: 4
POS: 11 NODE: 12
POS: 12 NODE: 10

REVERSED ORDERED NODE: 10
DFS DISCOVERY: 10
DTIME: 13
COMP: 0
DFS CLOSING: 10
FTIME: 13
REVERSED ORDERED NODE: 12
DFS DISCOVERY: 12
DTIME: 14
COMP: 1
DFS CLOSING: 12
FTIME: 14
REVERSED ORDERED NODE: 4
DFS DISCOVERY: 4
DTIME: 15
COMP: 2
KID: 7 PRED: 4
DFS DISCOVERY: 7
DTIME: 15
COMP: 2
KID: 11 PRED: 7
DFS DISCOVERY: 11
DTIME: 15
COMP: 2
DFS CLOSING: 11
FTIME: 15
DFS CLOSING: 7
FTIME: 16
DFS CLOSING: 4
FTIME: 17
REVERSED ORDERED NODE: 11
REVERSED ORDERED NODE: 7
REVERSED ORDERED NODE: 3
DFS DISCOVERY: 3
DTIME: 18
COMP: 3
DFS CLOSING: 3
FTIME: 18
REVERSED ORDERED NODE: 0
DFS DISCOVERY: 0
DTIME: 19
COMP: 4
DFS CLOSING: 0
FTIME: 19
```

```
REVERSED ORDERED NODE: 2
DFS DISCOVERY: 2
DTIME: 20
COMP: 5
KID: 5 PRED: 2
DFS DISCOVERY: 5
DTIME: 20
COMP: 5
DFS CLOSING: 5
FTIME: 20
KID: 6 PRED: 2
DFS DISCOVERY: 6
DTIME: 21
COMP: 5
KID: 9 PRED: 6
DFS DISCOVERY: 9
DTIME: 21
COMP: 5
KID: 8 PRED: 9
DFS DISCOVERY: 8
DTIME: 21
COMP: 5
KID: 1 PRED: 8
DFS DISCOVERY: 1
DTIME: 21
COMP: 5
DFS CLOSING: 1
FTIME: 21
DFS CLOSING: 8
FTIME: 22
DFS CLOSING: 9
FTIME: 23
DFS CLOSING: 6
FTIME: 24
DFS CLOSING: 2
FTIME: 25
REVERSED ORDERED NODE: 5
REVERSED ORDERED NODE: 1
REVERSED ORDERED NODE: 8
REVERSED ORDERED NODE: 9
REVERSED ORDERED NODE: 6
```

```
SCC:
NODE: 0 COMP: 4
NODE: 1 COMP: 5
NODE: 2 COMP: 5
NODE: 3 COMP: 3
NODE: 4 COMP: 2
NODE: 5 COMP: 5
NODE: 6 COMP: 5
NODE: 7 COMP: 2
NODE: 8 COMP: 5
NODE: 9 COMP: 5
NODE: 10 COMP: 0
NODE: 11 COMP: 2
NODE: 12 COMP: 1

COMP:
NODE: 10 COMP: 0

NODE: 12 COMP: 1

NODE: 4 COMP: 2
NODE: 7 COMP: 2
NODE: 11 COMP: 2

NODE: 3 COMP: 3

NODE: 0 COMP: 4

NODE: 1 COMP: 5
NODE: 2 COMP: 5
NODE: 5 COMP: 5
NODE: 6 COMP: 5
NODE: 8 COMP: 5
NODE: 9 COMP: 5
```