TP 3

Exercice 1 :

```
        System.out.println("Exercice 1, matrice G2 :");
        float[][] mat = graphM.getAdjmat();
        for(int i=0; i<mat.length; i++) {
            for(int j=0; j<mat.length; j++) {
                if(mat[i][j] != 0) {
                    System.out.printf("(%d,%d)", i, j);
                }
            }
        }
        System.out.println();
```

```
10 : 9
Exercice 1, matrice G2 :
(0,1)(0,2)(1,6)(2,1)(2,5)(3,2)(3,7)(4,3)(5,4)(5,7)(7,1)(7,2)(8,9)
If we choose the representation by adjacency lists
```

```
10 : 9
Exercice 1, matrice G1 :
(0,1)(0,2)(2,1)(2,5)(3,7)(4,3)(5,4)(5,7)(7,1)(7,2)(8,9)
```

Exercice 2 :

```
    /**
     * @param outMatrix display the result as an adjacency matrix (or as an
adjacency list)
     */
    public void printTransposed(boolean outMatrix) {
        if (outMatrix)
            Tools4A.printMatrix(transposedMM());
        else
            Tools4A.printAdjList(transposedML(), transposedMLW());
    }


    /**
     * Compute the transposed graph, represented by an unweighted adjacency list
     */
    private Node4A[] transposedML() {
        if (this.weighted == 1) return null;

        Node4A[] adjlist = new Node4A[this.n];
        //liste des successeurs du noeud j
        for (int j = 0; j < this.n; j++) {
            int[] successeurs = new int[this.n];
            int k = 0;
            for (int i = 0; i < this.n; i++) {
                if (this.adjmat[i][j] != 0) {
                    successeurs[k] = i;
                    k++;
                }
```

```
            }
            //ajout du noeud j et de ses successeurs dans adjlist
            Node4A n = new Node4A(successeurs[0], null); //n.val = premier
successeur de n
            adjlist[j] = n;
            k = 1;
            while (successeurs[k] != 0) { //n.next = autre successeur de n
                Node4A s = new Node4A(successeurs[k], null);
                n.setNext(s);
                n = s;
                k++;
            }
        }
        return adjlist;
    }
```

Complexité : $0(n^2)$

Exercice 3 :

```
    /**
     * TP2
     * Perform a graph search using the DFSnum algorithm
     */
    public void search() {
        this.debut = new int[this.n];
        this.fin = new int[this.n];
        this.arcType = new int[this.n][this.n];
        this.cycle = new LinkedList<Integer>();
        for (int i=0; i<this.n; i++) { //parcours de tous les noeuds (juste les noeuds, pas leurs
successeurs)
            if (this.debut[i] == 0) {
                this.nb += 1;
                this.debut[i] = this.nb;
                this.cycle.add(i);
                System.out.print(i+1 + " ");
                if (this.weighted == 0)
                    DFS_Num(adjlist[i], i);
                else
                    DFS_NumW(adjlistW[i], i);
            }
        }
        System.out.println();


    /**
     * TP2
     * @param s the vertex root of the tree provided by the DFSnum algorithm
     * @param sval the value of the node s (on n'y a pas vraiment accès si on le garde pas)
     */
    private void DFS_Num(Node4A s, int sval) {
        //récursivité sur chaque successeur de s
        for (Node4A next = s; next != null; next = next.getNext()){
               if (this.debut[next.getVal()] == 0) {
                //tree arc
                this.nb += 1;
                this.debut[next.getVal()] = this.nb;
                this.arcType[sval][next.getVal()] = 1;
                this.cycle.add(next.getVal());
                System.out.print(next.getVal()+1 + « »);

                      DFS_Num(adjlist[next.getVal()], next.getVal());

               }
            else {
```

```
                //not tree arc
                if (this.fin[next.getVal()] == 0) {
                    if (this.debut[sval] < this.debut[next.getVal()])
                        this.arcType[sval][next.getVal()] = 2; //forward (d[s]<d[n]) arc
                    else {
                        this.arcType[sval][next.getVal()] = 3; //backward (d[s]>d[n]) arc (=cycle)
                        //tous les noeuds entre s et n dans this.cycle sont dans le cycle
                        //System.out.println("Présence d'un cycle :");
                        //for(int i=this.cycle.indexOf(next.getVal()); i<this.cycle.indexOf(sval)+1;
i++)
                        //    System.out.print(this.cycle.get(i) + 1 + " ");
                        //System.out.println();
                    }
                }
                else
                    this.arcType[sval][next.getVal()] = 4; //cross arc
            }
        }
        this.nb += 1;
        this.fin[sval] = this.nb;
        this.cycle.remove((Integer) sval);
    }
```

main :

```
            System.out.println("Exercice 3 : ");
               float[][] mat = graphM.getAdjmat();
               for(int i=0; i<mat.length; i++) {
                     for(int j=0; j<mat.length; j++) {
                            if(mat[i][j] == 1) {
                                   System.out.printf("(%d,%d) de type tree arc\n", i, j);
                            }
                            if(mat[i][j] == 2) {
                                   System.out.printf("(%d,%d) de type forward arc\n", i, j);
                            }
                            if(mat[i][j] == 3) {
                                   System.out.printf("(%d,%d) de type backward arc\n", i, j);
                            }
                            if(mat[i][j] == 4) {
                                   System.out.printf("(%d,%d) de type cross arc\n", i, j);
                            }
                     }
               }
               System.out.println();
```

```
DFSNum(s) + cycle search.
Exercice 3
1 2 7 3 6 8 5 4 9 10
Début et fin de chaque sommet :
```

```
0  0  0  0  0  0  0  0
Exercice 3 :
(0,1) de type tree arc
(0,2) de type tree arc
(1,6) de type tree arc
(2,1) de type tree arc
(2,5) de type tree arc
(3,2) de type tree arc
(3,7) de type tree arc
(4,3) de type tree arc
(5,4) de type tree arc
(5,7) de type tree arc
(7,1) de type tree arc
(7,2) de type tree arc
(8,9) de type tree arc
```

Exercice 4 :

```
Exercice 4
Sommet 0 : poids=7
Sommet 1 : poids=16
Sommet 2 : poids=14
Sommet 3 : poids=6
Sommet 4 : poids=2
Sommet 5 : poids=7
Sommet 6 : poids=4
Sommet 7 : poids=6
Sommet 8 : poids=2
Sommet 9 : poids=0
------------------------------------------------------------
```

Main :

```
System.out.println("Exercice 4");
graphL.SommePoidsSommets();
```

GraphL4A :

```
private int[] somme;


public void SommePoidsSommets() {
   this.somme = new int[this.n];

   for (int i=0; i<this.n; i++) {
          for (WeightedNode4A next = adjlistW[i]; next != null; next = next.getNext()) {
                 somme[i] += next.getWeight();
          }
          System.out.printf("Sommet %d : poids=%d\n", i, this.somme[i]);
   }

}
```