INPT – ENSEEIHT Anthony Foulfoin & Ahmed Reda Boukhari

Département IMA 1ère année

Groupe B

**PROJET LONG D’ALN**

**Réduction d’une matrice creuse sous forme triangulaire par blocs**

**Sommaire**

[1 Application de l’algorithme de Tarjan 2](#_Toc226543719)

[1.1 Exemple 1 2](#_Toc226543720)

[1.2 Exemple 2 4](#_Toc226543721)

[2 Exemples illustrant l’algorithme de Tarjan 5](#_Toc226543722)

[2.1 Exemple 1 5](#_Toc226543723)

[2.2 Exemple 2 6](#_Toc226543724)

[3 Structures de données 6](#_Toc226543725)

[4 Simplifications 10](#_Toc226543726)

[5 Exemples d’utilisation de la structure 11](#_Toc226543727)

[5.1 Exemple 1 11](#_Toc226543728)

[5.2 Exemple 2 14](#_Toc226543729)

# Application de l’algorithme de Tarjan

## Exemple 1

Commençons par le déroulement de l’algorithme sur le modèle de l’exemple 1 :

1

2

3

4

5

6

7

**Graphe orienté de l’exemple 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pile |  |  |  |  | 73 | 7 | 7 | 7 | 7 |  |  |  |  |
|  |  |  | 64 | 64 | 63 | 6 | 6 | 6 |  |  |  |  |
|  |  | 5 | 5 | 5 | 5 | 53 | 5 | 5 |  |  |  |  |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 43 | 4 |  |  | 2 |  |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | vide | 1 | 1 | 1 |
| Etapes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

**L’état de la pile pendant le déroulement de l’algorithme de Tarjan**

Généralement ce cas est analogue a celui décrit dans l’exemple en dehors du fait que dans ce cas nous commencerons par un nœud qui possède un nœud entrant.

On commence par mémoriser 2. Et jusqu’à l’étape 5, on ne fait que mémoriser le chemin 3->4->5->6. À l’étape, le nœud 6 pointe vers un nœud déjà visité, le nœud 4, on met donc à jour son indice. À l’étape 5 on mémorise 7 on détecte un lien vers le nœud 3. On élimine 7 du chemin et le nœud 6 hérite donc du lien vers le nœud 7. On continue ainsi la propagation du lien vers le nœud 3 jusqu’a l’étape 9, on élimine alors tout le chemin 3->4->5->6->7. Notre pile est alors vide à l’étape 10, or on constate que le nœud 1 et 2 ne sont pas encore visités, on mémorise alors le nœud 1, et à l’étape 12 on mémorise le nœud 2, et, puisqu’il n’y a plus de chemins sortants du nœud 2, et que tous les nœuds ont été visités, on élimine 2 du chemin à l’étape 13.

## Exemple 2

3

1

2

4

5

7

8

6

**Graphe orienté de l’exemple 2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pile |  |  |  |  |  |  |  |  |  | 8 |  |  |  |  |
|  |  |  |  |  |  |  |  | 7 | 7 | 76 | 7 | 7 |  |
|  |  |  |  |  |  |  | 64 | 6 | 6 | 6 | 6 | 6 |  |
|  |  |  |  |  |  | 5 | 5 | 54 | 54 | 54 | 54 | 5 |  |
|  |  |  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |
|  |  | 12 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 3 | 3 | 32 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Etapes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**L’état de la pile pendant le déroulement de l’algorithme de Tarjan**

A l’étape 2 on mémorise le chemin 2->3, on empile donc ces 2 valeurs. À l’étape 3, on mémorise le nœud 1, et on détecte un lien vers le nœud 2. À l’étape 4 on élimine alors le nœud 1 et le nœud 3 hérite le lien vers le nœud 2. Ensuite à l’étape 5 on élimine le chemin 3->1, mais on conserve le cycle 1->2->3->1 car le nœud 2 possède encore un arc sortant vers un autre nœud 4 qui est encore non visité ; à l’étape 7 on mémorise alors le chemin 4->5, à l’étape 8 on mémorise le nœud 6 et on détecte un cycle (le lien 4), le lien vers le nœud 4 se propage alors vers le nœud 5 et on mémorise le nœud 7. À l’étape 9 on mémorise le chemin 7->8. Pour les autres étapes on procède par analogie à l’exemple fournie.

# Exemples illustrant l’algorithme de Tarjan

## Exemple 1

Dans ce cas on traite le cas ou l’on a 2 blocs séparés :

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PILE |  |  |  |  |  |  | 6 |  |  |  |  |
|  |  |  |  |  | 4 | 4 | 4 |  |  |  |
| 1 | 2 | 3 |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| étapes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  |  |  |  |  |  |  |  |  |  |  |  |

1

2

3

5

6

4

## Exemple 2

Voici un autre exemple illustrant l’algorithme de Tarjan :

4

1

2

3

6

5

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PILE |  |  |  |  |  | 5 |  | 6 |  |  |
|  |  |  | 42 | 4 | 4 | 4 | 4 | 4 |  |
|  |  | 3 | 3 | 32 | 32 | 32 | 32 | 3 |  |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| étapes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# Structures de données

Afin d’implémenter l’algorithme de Tarjan au sein de fortran, nous avons choisis d’utiliser 3 tableaux différentes, basées sur des tableaux.

1èr tableau : La pile de base de l’algorithme. Il s’agit de la pile décrite dans l’algorithme de Tarjan. Afin de mémoriser toutes les informations nécessaires à l’exécution de l’algorithme, nous y empilerons une structure composée de 3 éléments : le numéro du nœud, l’indice du nœud, et un booléen indiquant si le nœud fait partie ou non du chemin courant (équivalent de la « boite » décrite dans le sujet).

Numéro : numéro du nœud

Indice : indice du nœud

Chemin : le nœud fait partie du chemin ?

**Structure empilée dans la principale pile du programme**

Ainsi, la pile principale du programme contiendra une succession de structures telle que définie précédemment.

Numéro du nœud

Indice du nœud

Le nœud fait partie du chemin ?

Numéro du nœud

Indice du nœud

Le nœud fait partie du chemin ?

**Pile principale utilisant la structure définie précédemment**

Ainsi, avec cette structure, si le graphe se compose de N nœud, le programme consommera alors 3\*N d’espace mémoire.

Il reste 2\*N d’espace mémoire libre. Nous allons utiliser cet espace mémoire afin de stocker des informations supplémentaires au sein de deux tableaux de taille N, nous permettant ainsi de simplifier l’algorithme final.

2ème tableau : Le second tableau est un tableau de taille N, celui-ci contient la liste des tous les nœuds visités. Ainsi, un simple parcourt linéaire de ce tableau nous permet de savoir si un nœud à déjà été visité ou non.

Arrivé à cette étape, le programme consomme 3\*N + N = 4\*N d’espace mémoire.

3ème tableau : Le troisième et dernier tableau, de taille N, permet d’associer à chaque nœud le nombre d’arrêtes parcourues qui lui sont adjacentes. Pour un nœud p, cette information sera stockée à l’indice p. Ainsi, si le programme traite un nœud, celui-ci consultera tout d’abord ce tableau afin de savoir s’il reste ou non des arrêtes non parcourues. Si le nombre d’arrêtes parcourues est égal au nombre d’arrêtes sortantes du nœud (nombre que nous connaissons), alors toutes les arrêtes ont été visitées. Dans le cas contraire, le nombre stocké dans le tableau permet d’indiquer au programme quelle arrête parcourir.

Description de l’algorithme :

Soient :

* tab\_visites le tableau mémorisant tous les nœuds visités
* tab\_arretes le tableau mémorisant pour chaque nœud le nombre d’arrêtes parcourues.
* Pile la pile principale du programme
* Nb le nombre d’éléments stockés dans la pile à un moment donné.
* G le graph
* N le nombre de nœuds du graph
* Perm le tableau contenant les permutations
* NbCf la variable indiquant le nombre de composantes fortes détectées
* PtDebCF le tableau qui pour chaque composante forte de Perm, pointe sur sa première valeur

**1.** L’algorithme boucle tant que Pile n’est pas vide et qu’il reste des nœuds non visités.

A une étape donnée, l’algorithme cherche dans G une arête non encore parcourue adjacente au dernier nœud du chemin courant Pile[p] (soit pi ce nœud p d’indice i, i pouvant être égal à p, i étant sinon par construction toujours un nœud situé plus bas que p dans la pile).

Si la pile est vide, alors on choisit le premier nœud de G qui ne figure pas dans tab\_visites et on l’empile.

Sinon, l’algorithme consulte alors tab\_arretes.

**1.1** Si tab\_arretes[p] est égal à G[p]->indice, alors il n’existe pas d’arête non parcourue adjacente à pi.

**1.1.1** Si i est plus bas dans Pile que p, alors p est laissé dans Pile mais est enlevé du chemin courant : Pile[p]->chemin = faux.

Soit kr le nœud k de lien r précédant p dans le chemin courant (k = Pile[p-1]). Ce nœud existe et devient donc le dernier nœud du chemin. Le lien du nœud k devient soit r (Pile[k]->indice = r, si le nœud r est plus bas dans la pile que le nœud i), soit i (Pile[k]->indice = i, si le nœud i est plus bas dans la pile que le nœud r).

**1.1.2** Si i = p, alors dans ce cas le nœud p est enlevé du chemin courant (Pile[p]->chemin = faux). De plus p ainsi que tous les nœuds au dessus de lui dans la pile constituent une composante forte que l’on peut enlever de la pile. On dépile alors tous les éléments de Pile entre les indices Nb et p compris. On empile ensuite ces éléments dans Perm, puis on incrémente NbCf. On ajoute ensuite dans PtDebCF[NbCf] l’indice dans Perm du premier élément de la composante forte.

**1.2** Sinon, si tab\_arretes[p] < G[p]->indice, alors on consulte l’arrête d’indice tab\_arretes[p]+1 dans la liste d’adjacence de p. Puis, on incrémente le nombre d’arrêtes parcourues : tab\_arretes[p+1]++. Soit j le nœud pointé par l’arrête. Si j n’est pas présent dans tab\_visites, on l’y ajoute.

**1.2.1** Si j n’est pas dans la pile, alors on l’ajoute au sommet de la pile et au chemin courant : Pile[Nb+1] = (j,j,vrai). Puis on incrémente Nb. Enfin, on ajoute j dans tab\_visites.

**1.2.2** Si j est plus bas dans la pile que Pile[p]->indice, alors le lien est mis à jour pour pointer vers ce nouveau lien Pile[p]->indice = j.

**1.2.3** Si j est plus haut dans la pile que Pile[p]->indice, ou si j est présent dans Perm, on ne fait rien.

# Simplifications

Une matrice sous forme triangulaire est représentée par un graphe qui ne contient aucun cycle. Par conséquent, aucun nœud ne possède d’arrête pointant sur un nœud le précédant dans le graph. L’attribut indice de la structure définie dans la partie 3 devient ainsi inutile. Nous économisons ainsi N d’espace mémoire.

Quelques simplifications peuvent également être effectuées au niveau de l’algorithme :

* Dans le cas 1.1 de l’algorithme décrit ci-dessus, l’indice n’intervient pas. Le nœud courant est directement dépilé pour être empilé dans le tableau des permutations.
* Dans le cas 1.2. Si l’on se trouve dans la condition 1.2.1, alors l’indice n’est pas renseigné, étant donné que l’attribut indice est inutile ici. Les conditions 1.2.2 et 1.2.3 quant à elles n’ont plus lieu d’être. Si un nœud pointe vers un nœud plus haut que lui dans la pile, alors la matrice n’a pas de forme triangulaire.

# Exemples d’utilisation de la structure

## Exemple 1

1

2

3

4

5

6

7

**Graphe orienté de l’exemple 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pile  Avec la nouvelle structure |  |  |  |  | True | False | False | False | False |  |  |  |  |
|  |  |  |  | 3 | 7 | 7 | 7 | 7 |  |  |  |  |
|  |  |  |  | 7 | 7 | 7 | 7 | 7 |  |  |  |  |
|  |  |  | True | True | True | False | False | False |  |  |  |  |
|  |  |  | 4 | 4 | 3 | 6 | 6 | 6 |  |  |  |  |
|  |  |  | 6 | 6 | 6 | 6 | 6 | 6 |  |  |  |  |
|  |  | True | True | True | True | True | False | False |  |  |  |  |
|  |  | 5 | 5 | 5 | 5 | 3 | 5 | 5 |  |  |  |  |
|  |  | 5 | 5 | 5 | 5 | 5 | 5 | 5 |  |  |  |  |
|  | True | True | True | True | True | True | True | False |  |  | False |  |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 4 |  |  | 2 |  |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |  | 2 |  |
| True | True | True | True | True | True | True | True | True |  | True | True | True |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |  | 1 | 1 | 1 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |  | 1 | 1 | 1 |
| Etapes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

**L’état de la pile pendant le déroulement de l’algorithme de Tarjan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Indice |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Tab\_visites | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 7 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 6 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |  |
| 5 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 4 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 3 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |  |
| Etapes |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**L’état de tab\_visites pendant le déroulement de l’algorithme de Tarjan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Indice |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Tab\_noeuds | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 |  |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  |
| 5 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |  |
| 4 | 0 | 0 | 0 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |  |
| 3 | 0 | 0 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |  |
| 2 | 0 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |
| 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |  |
| Etapes |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**L’état de tab\_noeuds pendant le déroulement de l’algorithme de Tarjan**

La pile principale reste identique à celle illustrée dans la partie 1.1. Mais les boites et les indices sont maintenant représentés par un attribut au sein d’une structure, en plus du numéro du nœud.

A chaque étape, tab\_noeud mémorise les nœuds qui ont été visités pendant l’exécution de l’algorithme. Si on traite un nœud qui ne figure pas dans ce tableau, on l’y ajoute.

A chaque fois que l’on traite une nouvelle arrête sortante d’un nœud, alors on incrémente tab\_arrêtes à l’indice correspondant au nœud en question.

## Exemple 2

3

1

2

4

5

7

8

6

**Graphe orienté de l’exemple 2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pile |  |  |  |  |  |  |  |  |  | True |  |  |  |  |
|  |  |  |  |  |  |  |  |  | 8 |  |  |  |  |
|  |  |  |  |  |  |  |  |  | 8 |  |  |  |  |
|  |  |  |  |  |  |  |  | True | True | True | False | False |  |
|  |  |  |  |  |  |  |  | 7 | 7 | 6 | 7 | 7 |  |
|  |  |  |  |  |  |  |  | 7 | 7 | 7 | 7 | 7 |  |
|  |  |  |  |  |  |  | True | True | True | False | False | False |  |
|  |  |  |  |  |  |  | 4 | 6 | 6 | 6 | 6 | 6 |  |
|  |  |  |  |  |  |  | 6 | 6 | 6 | 6 | 6 | 6 |  |
|  |  |  |  |  |  | True | True | True | True | True | True | False |  |
|  |  |  |  |  |  | 5 | 5 | 4 | 4 | 4 | 4 | 5 |  |
|  |  |  |  |  |  | 5 | 5 | 5 | 5 | 5 | 5 | 5 |  |
|  |  |  |  |  | True | True | True | True | 4 | True | 4 | True |  |
|  |  |  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |
|  |  |  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |
|  |  | True | True | True | True | True | True | True | True | True | True | True | True |
|  |  | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  | 1 | 1 |
|  |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | True | True | True | True | True | True | True | True | True | True | True | True | True |
|  | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| True | True | True | True | True | True | True | True | True | True | True | True | True | True |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Etapes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**L’état de la pile pendant le déroulement de l’algorithme de Tarjan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Indice |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Tab\_visites | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |  |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |  |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 2 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |  |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| Etapes |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**L’état de tab\_visites pendant le déroulement de l’algorithme de Tarjan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Indice |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Tab\_noeuds | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 8 | 8 | 8 | 8 |  |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 | 7 | 7 |  |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |  |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |  |
| 4 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 2 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |  |
| 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |  |
| Etapes |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**L’état de tab\_noeuds pendant le déroulement de l’algorithme de Tarjan**

La pile principale reste identique à celle illustrée dans la partie 1.1. Mais les boites et les indices sont maintenant représentés par un attribut au sein d’une structure, en plus du numéro du nœud.

A chaque étape, tab\_noeud mémorise les nœuds qui ont été visités pendant l’exécution de l’algorithme. Si on traite un nœud qui ne figure pas dans ce tableau, on l’y ajoute.

A chaque fois que l’on traite une nouvelle arrête sortante d’un nœud, alors on incrémente tab\_arrêtes à l’indice correspondant au nœud en question.