Algorithmique 2 (INFO-F203)
Composantes fortement connexes

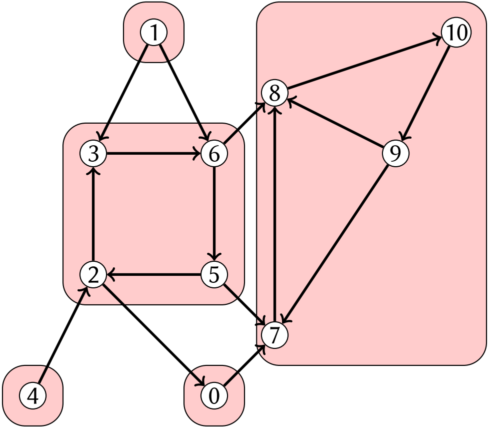
Jean Cardinal

Mars 2021

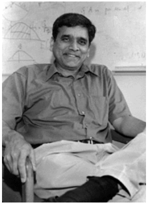
Connexité des graphes dirigés

- Un sommet u est dit accessible à partir d'un autre sommet v s'il existe un chemin *dirigé* de v vers u.
- Cette relation n'est pas symétrique : u peut être accessible à partir de v sans que v le soit à partir de u.
- Deux sommets u et v d'un graphe dirigé sont dits fortement connectés s'il existe un chemin dirigé de u vers v et également de v vers u.
- Cette relation est réflexive, symétrique et transitive, il s'agit donc d'une relation d'équivalence.

Composantes fortement connexes



Identification des composantes connexes



- Algorithme de Tarjan
- Algorithme de Kosaraju (John Hopkins)
- Redécouvert par Sharir (Tel-Aviv)

Algorithme de Kosaraju

- G^T est le *graphe transposé* de G, qui est obtenu en inversant l'orientation de chacun des arcs.
- L'algorithme de Kosaraju-Sharir consiste en les deux étapes suivantes :
  1. En utilisant un parcours en profondeur de G^T, identifier le postordre inverse de G^T.
  2. Effectuer un parcours en profondeur de G, en choisissant les sommets non-marqués dans le postordre inverse de G^T calculé à l'étape précédente.

Exemple

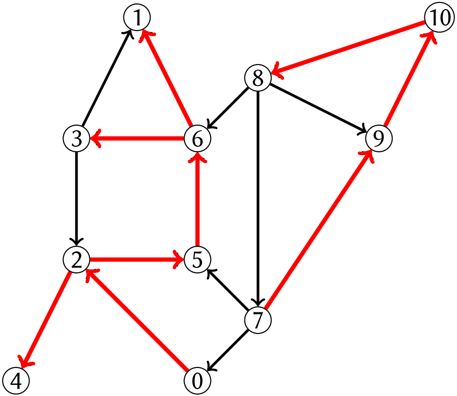


Illustration de la première phase de l'algorithme de Kosaraju-Sharir, sur le graphe transposé G^T : Deux appels à la procédure de parcours en profondeur donnent le préordre 0,2,4,5,6,1,3,7,9,10,8, le postordre 4,1,3,6,5,2,0,8,10,9,7, et le postordre inverse 7,9,10,8,0,2,5,6,3,1,4.

Exemple (2)

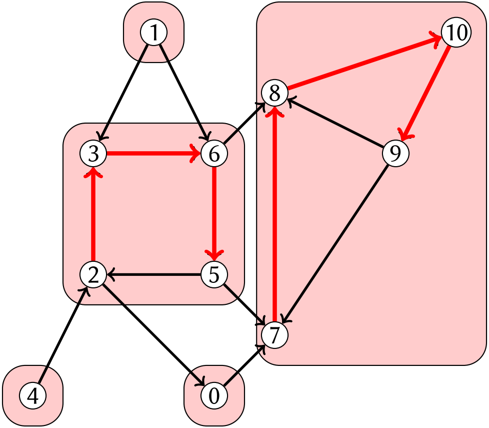


Illustration de la seconde phase de l'algorithme : Des appels du constructeur à la procédure de parcours en profondeur sont effectués sur les sommets 7,0,2,1,4, dans cet ordre. Les sommets accessibles forment les composantes fortement connexes.

Java

```
public KosarajuSharirSCC(Digraph G) {
  marked = new boolean[G.V()];
  id = new int[G.V()];
  DepthFirstOrder dfstree = new DepthFirstOrder(G.reverse());
  for (int v : dfstree.reversePostorder())
      if (!marked[v]) { dfs(G, v); count++; }
}
```

Proposition 1

L'algorithme de Kosaraju-Sharir construit les composantes fortement connexes du graphe.

Démonstration

1. Tous les sommets fortement connectés à `s` sont marqués lors de l'appel à `dfs(G,s)` dans le constructeur, lors du second parcours.
2. Tout sommet marqué lors de l'appel à `dfs(G,s)` dans le constructeur lors du second parcours est effectivement fortement connecté à `s`.

Première partie (fc $\Rightarrow$ marqué)

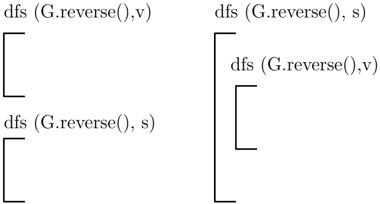
- On procède par contradiction, en supposant qu'il existe un sommet fortement connecté `v` qui n'est pas atteint lors de cet appel.
- Puisqu'il existe un chemin dirigé de `s` vers `v`, le sommet `v` doit avoir été marqué précédemment.
- Mais puisque par hypothèse il existe un chemin dirigé de `v` vers `s`, le sommet `s` aurait dû être marqué lors de l'appel `dfs(G,v)`, et le constructeur n'aurait jamais été appelé sur `s`, une contradiction.

Il est donc vrai que tout sommet fortement connecté à `s` est marqué lors de l'appel `dfs(G,s)` dans le constructeur.

Deuxième partie (marqué $\Rightarrow$ fc)

- Il existe donc un chemin dirigé de `s` vers `v`, et il reste à montrer qu'il existe un chemin dirigé de `v` vers `s`.
- De manière équivalente, nous devons montrer qu'il existe un chemin dirigé de `s` vers `v` dans G^T.
- **Observation** : Durant le *premier* parcours, la procédure `dfs(G.reverse(),v)` doit s'être terminée *avant* `dfs(G.reverse(),s)`, sans quoi `v` serait avant `s` dans le postordre inverse, et `v` aurait dû être traité avant `s` lors du second parcours.

Deux cas



Il reste donc deux cas à examiner. L'*appel* à `dfs(G.reverse(),v)` dans le premier parcours a été fait soit :

1. avant l'appel sur `dfs(G.reverse(),s)`
2. après l'appel à `dfs(G.reverse(),s)`.

Le premier cas est impossible, puisqu'on sait qu'il existe un chemin de `v` vers `s` dans G^T. Le second cas implique bien qu'il existe un chemin de `s` vers `v` dans G^T, ce qui conclut la démonstration.