Algorithmique des graphes - Cours 8

Olivier Baudon

Université de Bordeaux

21 octobre 2021

Correction DS 2020

Sujet

Voir
`https://dept-info.labri.fr/~baudon/Licence/Algo2/Cours/Annales/ds2020.pdf`

Exercice 1

Listes de successeurs - Algorithme

Données : G graphe orienté

ContientArcsOpposes(G) :

```
1:  pour tout u ∈ V(G) faire
2:      pour tout v ∈ Adj(u) faire
3:          pour tout w ∈ Adj(v) faire
4:              si w = u alors
5:                  retourner VRAI
6:              fin si
7:          fin pour
8:      fin pour
9:  fin pour
10: retourner FAUX
```

Exercice 1

Listes de successeurs - Complexité

Boucle 3-7 en $O(\Delta)$
Cette boucle est exécutée une fois par arc.
La boucle 1 passe en revue tous les sommets du graphe
Donc l'algorithme est en $O(n + m \times \Delta(G))$

Exercice 1

Matrice d'adjacence - Algorithme

Données : A, matrice d'adjacence d'un graphe G

ContientArcsOpposes(A) :

1: **pour** i de 1 à n **faire**
2: **pour** j de $i+1$ à n **faire**
3: **si** $A[i,j] = 1$ et $A[j,i] = 1$ **alors**
4: **retourner** VRAI
5: **fin si**
6: **fin pour**
7: **fin pour**
8: **retourner** FAUX

Exercice 1

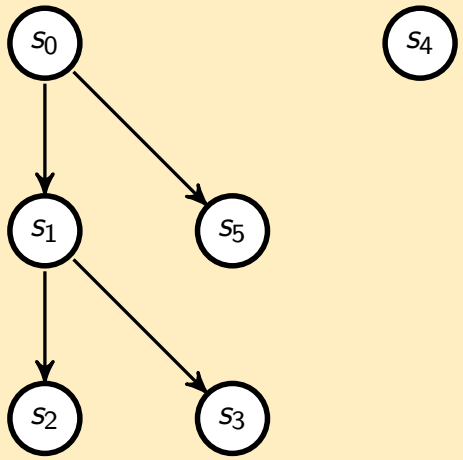
Matrice d'adjacence - Complexité

Dans le pire des cas (aucune paire d'arcs opposés), on parcourt tout le triangle de la matrice au dessus de la diagonale et donc on exécute $\frac{n \times (n-1)}{2}$ fois les lignes 3 à 5.
Par conséquent, la complexité est en $O(n^2)$.

Exercice 2

Question 1

Résultats du parcours en profondeur :

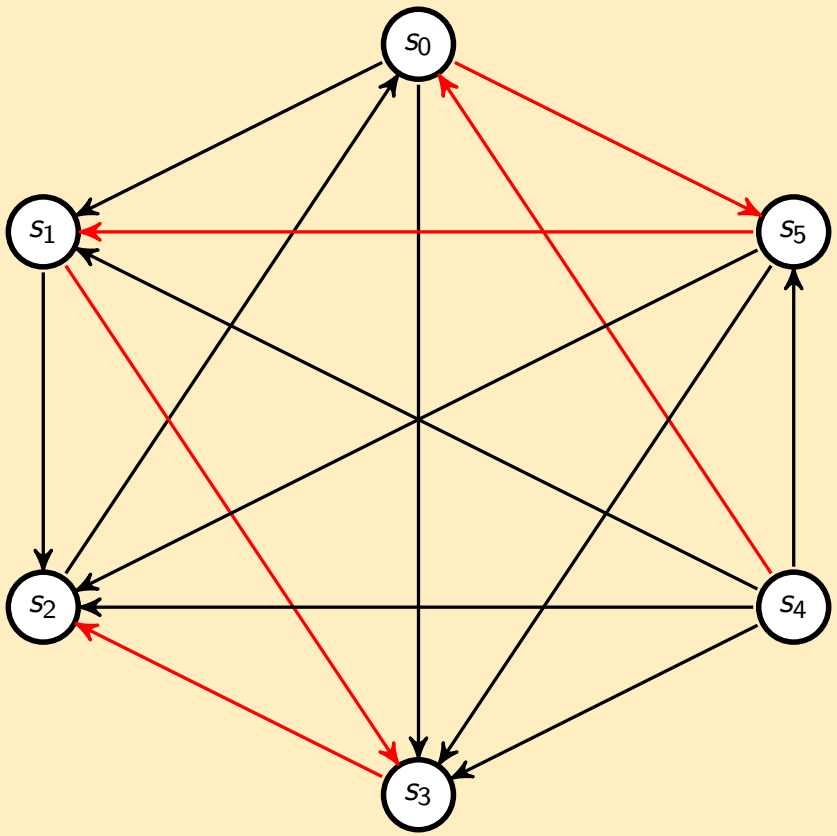


v	s_0	s_1	s_2	s_3	s_4	s_5
d(v)	1	2	3	5	11	8
f(v)	10	7	4	6	12	9
pere(v)	NIL	s_0	s_1	s_1	NIL	s_0

Exercice 2

Question 1 - suite

L'ordre inverse de fin de visite est : $s_4, s_0, s_5, s_1, s_3, s_2$. Cet ordre correspond à un chemin hamiltonien dans le tournoi.



Exercice 2

Question 2

Si v est le successeur de u dans l'ordre inverse de fin de visite, l'arc entre u et v sera soit un arc de liaison, soit un arc transverse.
Ça ne peut pas être un arc retour car dans ce cas $f(u) < f(v)$.
Donc u sera classé après v.
Ça ne peut pas être un arc avant car dans ce cas, il existe au moins un sommet w (par exemple le père de v dans l'arbre de liaison) tel que $f(u) > f(w) > f(v)$ et donc w serait situé entre u et v dans l'ordre inverse de fin de visite.

Exercice 2

Question 2 - suite

Soit u et v deux sommets successifs dans l'ordre inverse de fin de visite. On a $f(u) > f(v)$.
Si $d(u) < d(v)$, on est dans le cas d'un arc de liaison avec $d(u) < d(v) < f(v) < f(u)$ et dans ce cas, l'arc entre u et v est un arc de liaison de u vers v.
Si $d(u) > d(v)$, on est dans le cas d'un arc transverse avec $d(v) < f(v) < d(u) < f(u)$ et dans ce cas, l'arc entre u et v est un arc transverse de u vers v.
Donc dans les deux cas (arc de liaison ou arc transverse), l'arc entre u et v sera orienté de u vers v.

Exercice 2

Question 3

Il suffit d'afficher les sommets dans l'ordre inverse de fin de visite. Pour cela on utilise une pile où le premier élément empilé sera le premier sommet pour lequel l'appel de `Visiter_PP` se termine (il aura donc la plus petite valeur de `f[]`) et le dernier sommet empilé sera celui avec la plus grande valeur de `f[]`. À la fin on affiche les éléments de la pile dans l'ordre de dépilement. On rajoute donc :

- une ligne au début de `PP(G)` :

```
0 P <- Pile_Vide()
```

- une ligne à la fin de `Visiter_PP(u)` :

```
9 Empiler(P, u)
```

- une ligne à la fin de `PP(G)` :

```
8 Afficher(P)
```

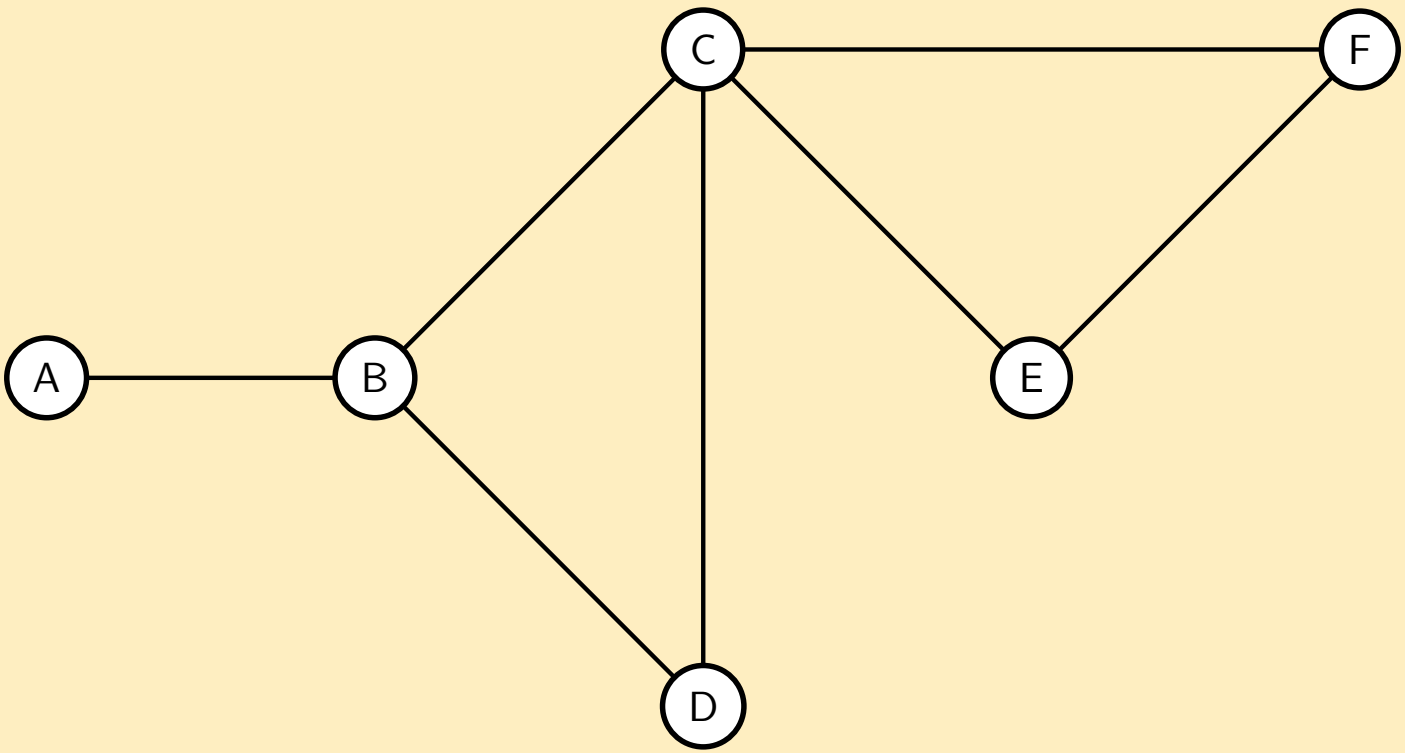
Exercice 2

Question 3

Aussi bien les opérations d'empilement que de dépilement s'effectuent en temps constant. Par conséquent, la complexité reste celle du parcours en profondeur, soit $O(n + m)$.

Exercice 3

Question 1



Exercice 3

Question 2

Il faut considérer les arêtes reliant deux sommets gris. Donc après la ligne 16, il faut rajouter :

```
17     sinon si couleur[v] = GRIS
18         alors cycle ← VRAI
```

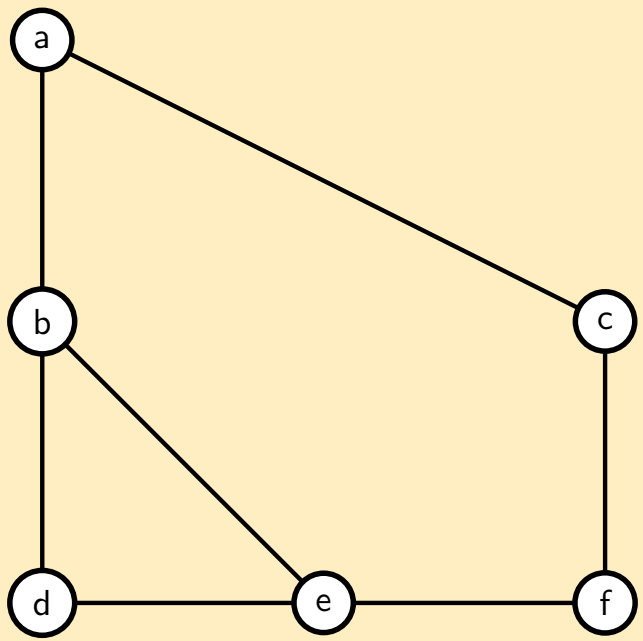
Question 3

La réponse est non. Il suffit de partir d'un sommet contenu dans aucun cycle, comme le sommet A de la question 3.1.

Exercice 3

Question 4

Dans l'exemple ci-dessous, on suppose que les sommets sont triés par ordre lexicographique et on part du sommet *a*. Le cycle *bdeb* sera détecté avant le cycle *abefca*.



Exercice 3

Question 5a

Sommet v	0	1	2	3	4	5	6	7	8
branche[v]	0	1	2	1	1	1	1	1	2

Exercice 3

Question 5b

PL-modifie(G, s) :

1: **pour tout** sommet u de $V(G)\backslash\{s\}$ **faire**
2: $\quad couleur[u] \leftarrow BLANC$
3: $\quad d[u] \leftarrow \infty$
4: $\quad pere[u] \leftarrow nil$
5: **fin pour**
6: $couleur[s] \leftarrow GRIS$
7: $d[s] \leftarrow 0$
8: $pere[s] \leftarrow nil$
9: $branche[s] \leftarrow 0$
10: $Enfiler(F, s)$

Exercice 3

Question 5b-suite

```
1: tant que nonvide(F) faire
2:     u ← tete(F)
3:     pour tout v de Adj(u) faire
4:         si couleur[v] = BLANC alors
5:             couleur[v] ← GRIS
6:             d[v] ← d[u] + 1
7:             pere[v] ← u
8:             Enfiler(F, v)
9:             si branche[u] = 0 alors
10:                branche[v] ← v
11:            sinon
12:                branche[v] ← branche[u]
13:            fin si
14:        sinon si couleur[v] = GRIS et branche[v] ≠ branche[u] alors
15:            retourner d[u] + d[v] + 1
16:        fin si
17:    fin pour
18:    Defiler(F)
19:    couleur[u] ← NOIR
20: fin tant que
21: retourner 0
```