Département Licence	Année : 2005/2006 Session 1 **PARCOURS :** INF 6 **UE :** INF 108, Algorithmes des graphes **Épreuve :** INF 108 CC **Date :** Samedi 29 avril 2006 **Heure :** 8 heures 30 **Durée :** 1 heure 30 Documents : non autorisés

SUJET + CORRIGE

Avertissement

- Les cinq exercices sont indépendants.
- Réponses directement sur la feuille. (Utilisez un brouillon pour éviter les ratures sur la copie).
- L'ensemble est assez long, mais le total des points est de 25.

Le sujet

Exercice 1 (Dénombrements (2 points)
Soit F une forêt (graphe simple non orienté sans cycle) de n sommets et m arêtes.
Quel est le nombre de composantes connexes de F ? Justifiez votre réponse.
Réponse :
Chaque composante connexe C_i de F est un arbre et vérifie $n_i - 1 = m_i$.
Soit k le nombre de composantes connexes. $\sum_{i=1}^{k}(n_i - 1) = \sum_{i=1}^{k} m_i \Rightarrow k = n - m$

Exercice 2 (Chemins infinis (4 points)
Rappels : *Après un parcours en profondeur, un arc (u, v) d'un graphe orienté G est :*
- *un arc de liaison ou un arc avant si $d[u] < d[v] < f[v] < f[u]$*
- *un arc retour si $d[v] < d[u] < f[u] < f[v]$*
- *un arc couvrant si $d[v] < f[v] < d[u] < f[u]$*

Un chemin infini est une suite infinie d'arcs de G non nécessairement distincts (s_i, s_{i+1}).

Dans un graphe orienté fini $G = (S, U)$, montrez que les propositions suivantes sont équivalentes :

P1 : G possède un arc retour après un parcours en profondeur.

P2 : G possède un circuit.

P3 : Il existe un chemin infini dans G.

Réponse :
- $P1 \Rightarrow P2$
  G possède un arc retour (u, v), il existe donc un chemin de liaison $v \rightsquigarrow u$ et par conséquent un circuit $v \rightsquigarrow u.(u, v)$.
- $P2 \Rightarrow P3$
  Il suffit d'itérer le circuit existant pour construire un chemin infini.
- $P3 \Rightarrow P1$
  Soit un chemin infini dans G et $f_0, f_1, \ldots, f_n, \ldots$ la suite des dates de fin correspondantes d'un parcours en profondeur. Cette suite infinie ne peut pas être décroissante, donc il existe $f_k > f_{k-1}$. L'arc correspondant est donc un arc retour.

Les trois propositions sont donc équivalentes.

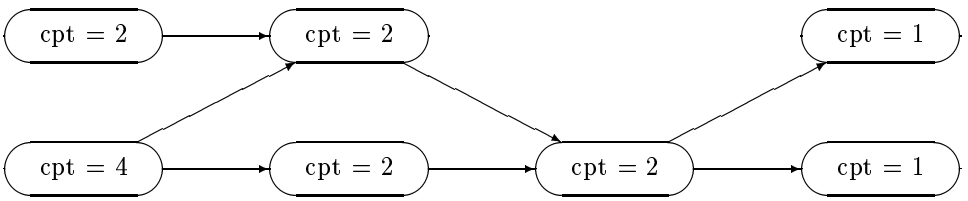
Exercice 3 (Parcours en profondeur (6 points))
Rappels : *Dans un graphe orienté $G = (S, U)$*
- *un chemin fini $s \rightsquigarrow t$ est une suite finie de n arcs $(s_i, s_{i+1}) \in U$ avec $s = s_0$ et $t = s_n$.*
- *un chemin élémentaire est un chemin fini tel que $\forall (i, j), s_i \neq s_j$*
- *un chemin élémentaire maximal à droite est un chemin élémentaire tel que $\forall x \in S, s \rightsquigarrow t.(t, x)$ n'est pas un chemin élémentaire.*
- *un chemin élémentaire maximal à gauche est un chemin élémentaire tel que $\forall x \in S, (x, s).s \rightsquigarrow t$ n'est pas un chemin élémentaire.*
- *un chemin élémentaire maximal est un chemin élémentaire maximal à droite et à gauche.*

Soit la variante suivante du parcours en profondeur.

```
ParcoursComptage(G)                          | Visiter_Comptage_PP(u)
1 pour chaque sommet u de U[G]               | 1 couleur[u] <- GRIS
2      faire couleur[u] <- BLANC             | 2 si Adj[u] = vide
3            cpt[u] <- 0                     | 3 alors cpt[u] <- 1
4 pour chaque sommet u de U[G]               | 4 sinon pour chaque v de Adj[u]
5      faire si couleur[u] = BLANC           | 5            faire si couleur[v] = BLANC
6            alors Visiter_Comptage_PP(u)    | 6                  alors Visiter_Comptage_PP(v)
                                             | 7                  cpt[u] <- cpt[u] + cpt[v]
                                             | 8 couleur[u] <- NOIR
```

Question 3.1 (1 point) *Exécutez l'algorithme sur ce graphe.*

Réponse :



Question 3.2 (1,5 point) *Que fait cet algorithme lorsqu'on l'applique à un graphe orienté simple et sans circuit* (À quoi correspond la valeur `cpt[u]` calculée) *?*

Réponse : `cpt[u]` *= nombre de chemins élémentaires maximaux à droite ayant u pour origine.*
L'initialisation à 1 correspond au chemin vide pour les sommets sans successeurs.

Question 3.3 (1,5 point) *En supposant qu'il existe une structure Pred[u] qui donne pour chaque sommet la liste de ses prédécesseurs, donnez un algorithme qui lorsqu'on l'applique à un graphe orienté simple et sans circuit calcule le nombre de chemins élémentaires maximaux.*

Réponse :

```
nbCheminsMaximaux(G)
1 ParcoursComptage(G)
2 total <- 0
3 pour chaque sommet u de U[G]
4      faire si pred[u] = vide
5            alors total <- total + cpt[u]
6 retourner total
```

Question 3.4 (2 points) *Quelle(s) modification(s) sont nécessaires pour traiter un graphe orienté sans arc multiple et sans circuit ?* (les boucles sur les sommets sont permises)

Réponse : *Dans* `nbCheminsMaximaux(G)` *la ligne*

```
4       faire si pred[u] = vide
```

est remplacée par

```
4'      faire si (pred[u] = vide ou pred[u] = {u})
```

et dans `Visiter_Comptage_PP(u)` *les lignes*

```
2 si Adj[u] = vide
7                   cpt[u] <- cpt[u] + cpt[v]
```

sont remplacées par

```
2' si (Adj[u] = vide ou Adj[u] = {u})
7'                  si (v != u)
7''                 alors cpt[u] <- cpt[u] + cpt[v]
```

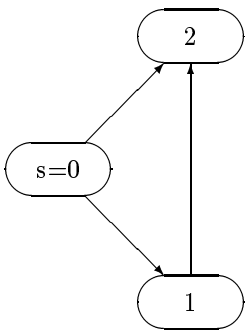
Exercice 4 (Comparaisons des parcours (6 points))

Question 4.1 (1 point) *Soit un arbre orienté $A = (S, U)$ et $r = racine(A)$. Montrez que les sous-graphes de liaison obtenus par les algorithmes* `ParcoursLargeur(A,racine(A))` *et* `ParcoursProfondeur(A,racine(A))` *sont identiques. (*`ParcoursProfondeur(G,s)` *est une variante qui consiste à ne visiter récursivement le graphe qu'à partir du sommet* `s`*).*

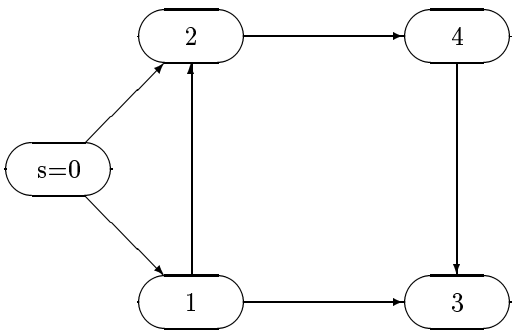
Réponse : *A est un arbre, et tous les sommets sont accessibles à partir de la racine. Les sous-graphes de liaison obtenus sont donc identiques car égaux à l'arbre A.*

Question 4.2 (1 point) *Donnez un exemple de graphe simple orienté acyclique $DAG = (S, U, s \in S)$ tel que les sous-graphes de liaison obtenus par* `ParcoursLargeur(DAG,s)` *et* `ParcoursProfondeur(DAG,s)` *soient distincts.*

Réponse :



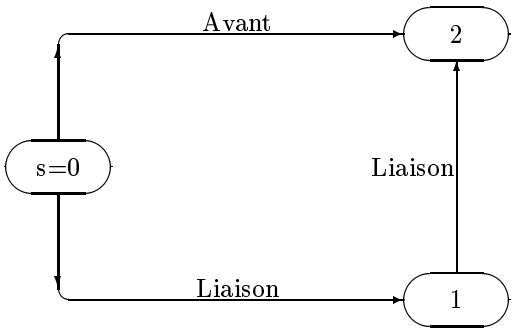
Ordre 0, 1, 2



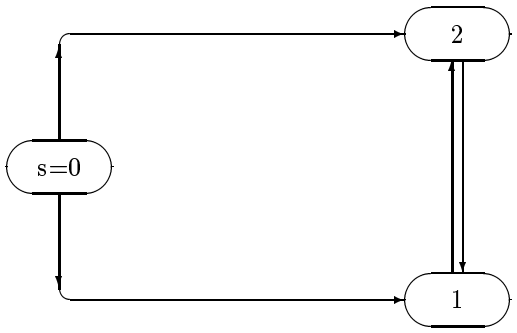
Pour tous les ordres

Question 4.3 (1 point) *Donnez un exemple de graphe simple orienté $G = (S, U, s \in S)$ tel que :*
- `ParcoursProfondeur(G,s)` *engendre au moins un arc avant,*
- *les sous-graphes de liaison obtenus par* `ParcoursLargeur(G,s)` *et* `ParcoursProfondeur(G,s)` *sont différents.*

Réponse : *Il suffit que l'arc avant diminue le chemin depuis s.*



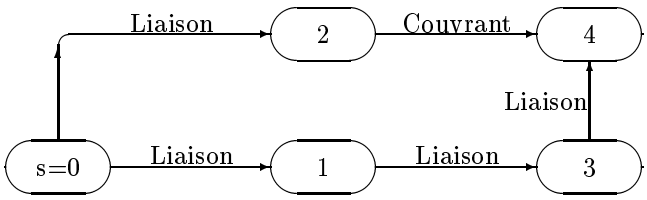
Ordre 0, 1, 2



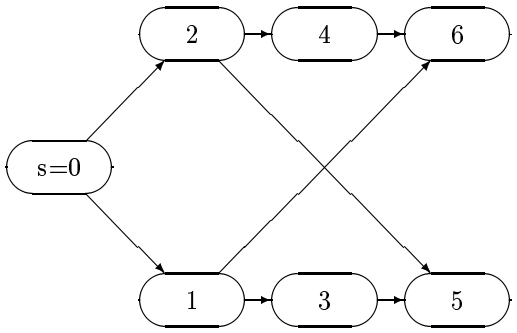
Tous les ordres ⇒ arc retour ou arc couvrant

Question 4.4 (1 point) *Donnez un exemple de graphe simple orienté $G = (S, U, s \in S)$ tel que :*
- `ParcoursProfondeur(G,s)` *engendre au moins un arc couvrant,*
- *les sous-graphes de liaison obtenus par* `ParcoursLargeur(G,s)` *et* `ParcoursProfondeur(G,s)` *sont différents.*

Réponse : *Il suffit que l'arc couvrant diminue le chemin depuis s.*



Ordre 0, 1, 2, 3, 4



Pour tous les ordres

Question 4.5 (2 points) *Soit un graphe simple orienté $G = (S, U, s \in S)$.*
Montrez que si `ParcoursProfondeur(G,s)` *engendre aucun arc avant et aucun arc couvrant, les sous-graphes de liaison obtenus par* `ParcoursLargeur(G,s)` *et* `ParcoursProfondeur(G,s)` *sont identiques.*

Réponse :
Soit l'ensemble des arcs de liaison de PL(G,s) qui ne sont pas des arcs de liaison de PP(G,s).
D'après l'hypothèse, tous ces arcs sont des arcs retour.
Soit (u, v) un tel arc avec $\delta(u)$ minimal. (distance de s).
(u, v) arc de liaison de PL(G,s) $\Rightarrow \delta(v) = \delta(u) + 1$.
(u, v) arc retour de PP(G,s) $\Rightarrow s \rightsquigarrow v \rightsquigarrow u$ dans le graphe de liaison de PP.
Le choix de (u, v) avec $\delta(u)$ minimal $\Rightarrow s \rightsquigarrow v \rightsquigarrow u$ également dans le graphe de liaison de PL.
On en déduit $\delta(v) < \delta(u)$. l y a donc contradiction.
L'ensemble des arcs de liaison de PL(G,s) qui ne sont pas des arcs de liaison de PP(G,s) est donc vide.
Les $n-1$ arcs de liaison de PL(G,s) sont donc des arcs de liaison de PP(G,s).
Il y a au maximum $n-1$ arcs de liaisons dans PP(G,s), les sous-arbres de liaison sont donc identiques.

Exercice 5 (Arbres couvrants (7 points))
On veut calculer X_n le nombre d'arbres couvrants de la grille $2 \times n$, c'est-à-dire le graphe $G_n = (\{x_{i,j} | 1 \leq i \leq 2,\ 1 \leq j \leq n\}, \{\{x_{1,j}, x_{2,j}\} |\ 1 \leq j \leq n\} \cup \{\{x_{i,j}, x_{i,j+1}\} | 1 \leq i \leq 2,\ 1 \leq j < n\})$.
On note également $H_n = G_n - \{\{x_{1,n}, x_{2,n}\}\}$, et Y_n le nombre d'arbres couvrants de H_n.

FIG. *1 – La grille G_n et son sous-graphe H_n*

Question 5.1 (1 point) *Quels sont X_1 et Y_1 ?*

Réponse :
$X_1 = 1$ et $Y_1 = 0$

Question 5.2 (1 point) *Démontrez que, pour tout $n > 1$, $Y_n = X_{n-1}$*

Réponse :
Le degré de $x_{1,n}$ et de $x_{2,n}$ dans H_n est toujours 1. Donc tout arbre couvrant de H_n contient les arêtes $\{x_{1,n-1}, x_{1,n}\}$ et $\{x_{2,n-1}, x_{2,n}\}$. Par conséquent un arbre couvrant de H_n diminué de ces deux arêtes est un arbre couvrant de G_{n_1} et réciproquement un arbre couvrant de G_{n-1} augmenté de ces deux arêtes est un arbre couvrant de H_n.
On en déduit que pour tout $n > 1$, $Y_n = X_{n-1}$.

Question 5.3 (1,5 point) *Soit $n > 1$ et T un arbre couvrant de la grille $2 \times n$, contenant les trois arêtes $\{x_{1,n}, x_{2,n}\}, \{x_{1,n-1}, x_{1,n}\}, \{x_{2,n-1}, x_{2,n}\}$.*
Montrez que T' obtenu à partir de T privé de ces 3 arêtes, et augmenté de l' arête $\{x_{1,n-1}, x_{2,n-1}\}$, est un arbre couvrant de la grille $2 \times (n-1)$.

Réponse :
T' n'a pas de cycle, sinon T aurait un cycle également.
T' est connexe sur la grille $2 \times (n-1)$.
T' possède $(2n-1) - 3 + 1 = 2(n-1) - 1$ arêtes.
T' est donc un arbre couvrant de la grille $2 \times (n-1)$.

Question 5.4 (1,5 point) *Exprimez, pour $n > 1$, le nombre Z_n d'arbres couvrants de la grille $2 \times n$, contenant les trois arêtes $\{x_{1,n}, x_{2,n}\}, \{x_{1,n-1}, x_{1,n}\}, \{x_{2,n-1}, x_{2,n}\}$, à partir des nombres X_{n-1} et Y_{n-1}.*

Réponse :
D'après la question précédente, on peut construire à partir de chaque arbre T un arbre couvrant distinct T' de G_{n-1} qui n'est pas un arbre couvrant de H_{n-1} car T' contient l'arête $\{x_{1,n-1}, x_{2,n-1}\}$.
On en déduit $Z_n = X_{n-1} - Y_{n-1} = X_{n-1} - X_{n-2}$

Question 5.5 (2 points) *Donnez une récurrence pour X_n $(n > 1)$.*

Réponse :
4 cas à considérer :
- *X_{n-1} arbres contiennent $\{x_{1,n}, x_{2,n}\}, \{x_{1,n-1}, x_{1,n}\}$*
- *X_{n-1} arbres contiennent $\{x_{1,n}, x_{2,n}\}, \{x_{2,n-1}, x_{2,n}\}$*
- *X_{n-1} arbres contiennent $\{x_{1,n-1}, x_{1,n}\}, \{x_{2,n-1}, x_{2,n}\}$*
- *$X_{n-1} - X_{n-2}$ arbres contiennent $\{x_{1,n}, x_{2,n}\}, \{x_{1,n-1}, x_{1,n}\}, \{x_{2,n-1}, x_{2,n}\}$*

On obtient $X_n = 4X_{n-1} - X_{n-2}$