

TD n°6 - Correction

Parcours de graphes

Exercice 2 Non ! Par exemple, le graphe de sommets $\{a, b, c, d, e\}$ et d'arêtes $\{(ab), (ac), (bd), (be), (cd), (ce)\}$. L'arbre d'arêtes $\{(ab), (ac), (bd), (ce)\}$ ne peut pas être obtenu par un parcours en largeur, mais est bien un arbre de plus courts chemins partant de a .

Exercice 4 On fait une modification du parcours en largeur. En effet, il trouve des plus courts chemins, donc il y a lieu d'espérer qu'il trouve aussi des plus courts circuits. Cet algorithme est suivi d'une preuve pour se convaincre de cette affirmation.

```
Procédure Cherche_Circuit(G, s)
// G = (S, A) graphe orienté
pour chaque x ∈ S faire
  Marquage[x] := faux ;
  Père[x] := nil ;
F := File_Vide ;
Enfiler(s,F) ;
Marquage[s] := vrai ;
Profondeur[s] := 0 ;
tant que F non vide faire
  x := Defiler(F) ;
  pour chaque y ∈ Voisinage(x, G) faire
    si non(Marquage[y]) alors
      Enfiler(y,F) ;
      Marquage[y] := vrai ;
      Père[y] := x ;
      Profondeur[y] := Profondeur[x] + 1 ;
    sinon
      si y = s alors
        tant que x ≠ nil faire
          afficher(x) ;
          x := Père[x] ;
        renvoyer vrai ;
```

Preuve :

- **Terminaison.** On étudie les exécutions possibles de l'algorithme. Si une exécution entre dans le sinon de la ligne 18, alors l'algorithme termine (en renvoyant vrai). Si au contraire elle n'entre jamais dans ce sinon, alors elle se déroule exactement comme un parcours en profondeur, et donc termine aussi. On conclut que l'algorithme termine.
- **Preuve que l'algorithme trouvera un circuit s'il y en a un passant par s.** On suppose qu'il existe au moins un circuit passant par s . On considère, parmi les circuits de longueur minimale passant par s , les sommets de plus grande profondeur en partant de s . Parmi ceux-là, l'un d'eux sera le premier

atteint par le parcours en largeur effectué par l'algorithme. Pour ce sommet, on entre dans le sinon de la ligne 18, et l'algorithme renvoie vrai.

Jusqu'à ce qu'on entre dans le sinon de la ligne 18, les invariants du cours sur le parcours en largeur sont vérifiés. En particulier, les sommets dont la distance depuis s est strictement inférieure ont déjà été visités, et l'arbre obtenu est un arbre des plus courts chemins depuis s .

- **Minimalité d'un circuit trouvé** Ainsi, au moment où on rentre dans le sinon, on trouve donc un circuit sur s . Et on sait que tout sommet x dont la distance à s est strictement inférieure a été visité précédemment. Puisqu'on n'est entré dans le sinon de la ligne 18 à la visite de x , c'est que x n'est pas lié à s . On en déduit que le cycle passant par s trouvé par l'algorithme est de longueur minimale.

Exercice 5 Tout graphe non orienté et connexe G vérifie une et une seule des conditions suivantes :

- soit G est un arbre,
- soit G contient un cycle.

Or d'après le cours, un graphe non orienté connexe $G = (S, A)$ est un arbre si et seulement si $|A| = |S| - 1$. Pour tester si un tel graphe possède un cycle, il suffit donc de tester si $|A| > |S| - 1$. Cet algorithme ne permet cependant pas d'exhiber un cycle dans le graphe.

Exercice 6 Un pseudo-code possible du parcours en profondeur avec relance. On n'utilise pas de marques : les dates remplacent. Un début à -1 signifie un sommet non encore atteint, un début ≥ 0 mais une fin de -1 signifie un sommet en cours de visite.

Variables globales: sommet x, y ; entier temps = 0, $d[V]$, $f[V]$ tableaux initialisés à -1

Pour x de 1 à n

 Si marquage[x] = faux
 PP(x)

Procédure PP(sommet v)

```
    d[v] = temps++
    pour chaque voisin  $y$  de  $v$  faire
        si d[y] == -1
            Afficher (v,y) est un arc de parcours
            PP(y)
        sinon si f[v] == -1
            Afficher (v,y) est un arc de retour (aussi appelé en arrière)
        sinon si f[y] < d[v]
            Afficher (v,y) est un arc transversal
        sinon
            Afficher (v,y) est un arc en avant
    f[v] = temps++
```