UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

LICENCE Informatique
Info0501 – 2020/2021

Travaux dirigés n° 2

Graphes - Algorithmes de parcours

Exercice 1 (Préliminaire - Piles)

1°) Rappelez le fonctionnement d'une pile d'au plus n éléments avec un tableau $p[1..n]$. Le tableau possède un attribut $p.sommet$ qui indexe l'élément le plus récemment inséré. La pile est constituée des éléments $p[1..p.sommet]$ où $p[1]$ est l'élément situé à la base de la pile et $p[p.sommet]$ l'élément l'élément situé au sommet. Donnez les trois états successifs suivants d'une pile d'au plus 7 éléments :

1. la pile après empilage successif des valeurs 15, 6, 2 et 9 ;
2. la pile après empilage successif des valeurs 17 et 3 ;
3. la pile après un dépilage.

2°) Définissez les procédures de pile suivantes :

a) `pile-vide` : teste si la pile est vide ;

b) `empiler` : empile une valeur dans la pile ;

c) `dépiler` : dépile un élément de la pile.

Exercice 2 (Préliminaire - Files)

1°) Rappelez le fonctionnement d'une file d'au plus $n-1$ éléments à l'aide d'un tableau $f[1..n]$. La file comporte un attribut $f.tete$ qui indexe vers sa tête où sera retiré le prochain élément. L'attribut $f.queue$ indexe le prochain emplacement où sera inséré un élément nouveau. Les éléments de la file se trouvent aux emplacements $f.tete$, $f.tete+1$, ... $f.queue-1$, après quoi l'on boucle : l'emplacement 1 suit immédiatement l'emplacement n dans un ordre circulaire. Quand $f.tete = f.queue$, la file est vide. Au départ, on a $f.tete = f.queue = 1$. Quand $f.tete = f.queue + 1$, la file est pleine. Donnez les trois états successifs suivants d'une file d'au plus 11 éléments :

1. la file où $f.tete = 7$, $f.queue = 12$ et où les valeurs, de la tête à la queue, sont 15, 6, 9, 8 et 4 ;
2. la file après l'enfilage successif des valeurs 17, 3 et 5 ;
3. la file après un défilage.

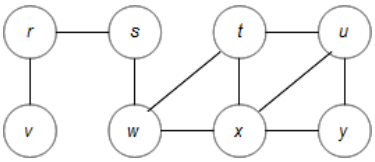
2°) Définissez les procédures de file suivantes :

a) `enfiler` : insère un élément dans la file ;

b) `défiler` : retire un élément de la file.

Exercice 3 (Parcours en largeur)

1°) Soit le graphe ci-contre, illustrez le fonctionnement d'un parcours en largeur démarrant au sommet s en donnant, pour chaque sommet, sa distance par rapport à s (le plus petit nombre d'arêtes comprises entre celui-ci et s). Donnez aussi l'arbre de parcours résultant.

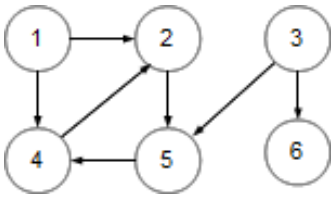


2°) Définissez une procédure `parcours-largeur` qui effectue le parcours en largeur d'un graphe G à partir de s et qui calcule la distance (plus petit nombre d'arcs) entre s et chaque sommet. Donnez-en le temps d'exécution.

3°) Définissez une procédure `afficher-chemin` qui affiche les sommets d'un plus court chemin reliant le sommet s à un sommet v de G, en supposant que `parcours-largeur` a déjà calculé un arbre de parcours en largeur. Donnez-en le temps d'exécution.

Exercice 4 (Parcours en profondeur)

1°) Soit le graphe ci-contre, illustrez le fonctionnement d'un parcours en profondeur en considérant les sommets dans l'ordre lexicographique et en donnant, pour chaque sommet, sa date de découverte et sa date de fin de traitement. Donnez aussi la forêt de parcours résultante.



2°) Définissez une procédure `parcours-profondeur-recursif` qui effectue le parcours en profondeur d'un graphe G de façon récursive et qui calcule, pour chaque sommet, les dates de découverte et de fin de traitement. Donnez-en le temps d'exécution.

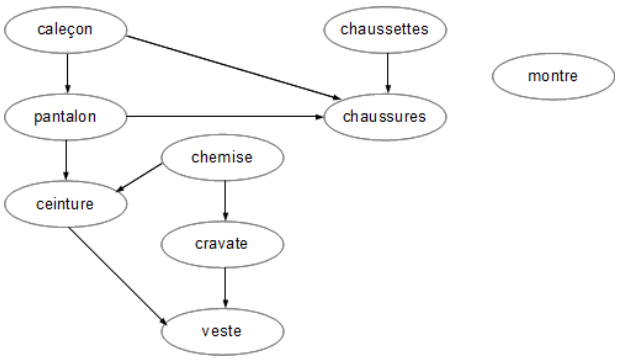
3°) Définissez une procédure `parcours-profondeur-itératif` qui effectue un parcours en profondeur d'un graphe G de façon itérative en se limitant à donner l'ordre de visite des sommets (il n'est pas nécessaire de calculer les dates de début, dates de fin et pères comme dans la version récursive). Donnez-en le temps d'exécution.

4°) Récrivez la procédure `parcours-profondeur-itératif` de sorte à obtenir un résultat identique à la version récursive, avec dates de début, dates de fin et arborescence de parcours identiques.

Exercice 5 (Tri topologique)

Les graphes acycliques orientés sont utilisés dans de nombreuses applications pour représenter des précédences entre événements. Le tri topologique d'un graphe acyclique orienté $G = (S, A)$ consiste à ordonner linéairement tous ses sommets de sorte que, si G contient un arc (u, v), u apparaisse avant v dans le tri.

La figure ci-contre donne un exemple de graphe acyclique. Celui-ci a été conçu par le savant professeur Cosinus qui avait de la difficulté à s'habiller le matin. Chaque arc (u, v) signifie que le vêtement u doit être enfilé avant le vêtement v. L'absence d'arc entre deux vêtements signifie que ceux-ci peuvent être enfilés dans n'importe quel ordre. À chaque matin, il réalise un tri topologique de ce graphe afin d'obtenir un ordre permettant de s'habiller correctement.



1°) Illustrez l'application d'un parcours en profondeur à ce graphe en notant, pour chaque sommet, sa date de découverte et sa date de fin de traitement. Déduisez-en un ordre permettant de s'habiller correctement.

2°) Définissez la procédure `tri-topologique` qui retourne une liste contenant tous les sommets triés topologiquement d'un graphe acyclique orienté. Donnez-en le temps d'exécution.

Références

Cormen T. H., Leiserson C. E., Rivest R. L. et Stein C., "Algorithmique" 3e édition, Dunod, 2010.