TD noté d'algorithmique

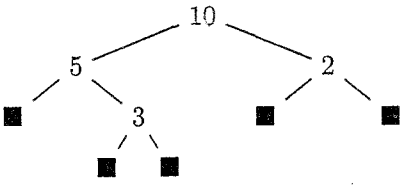
Mercredi 24 novembre 2010 10h30/12h30
Aucun document autorisé.

Exercice 1 : Arbres binaires de recherche – (8 points[1] :1/1/2/1/1/2)

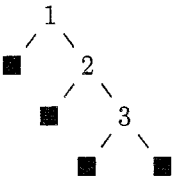
On va s'intéresser à la profondeur moyenne des nœuds dans un ABR. Deux précisions : (1) dans toutes ces questions, on ne s'intéressera pas à la valeur des clés contenues dans les nœuds internes ; (2) pour chaque question, il est possible de définir des fonctions supplémentaires si vous en avez besoin.

On rappelle que la profondeur d'un nœud s dans un ABR de racine a est la longueur du chemin reliant a à s (la profondeur de la racine est donc 0, celle de ses fils 1, etc.). Cette définition est valable pour les nœuds internes (avec une clé) et pour les nœuds externes (aussi appelés les feuilles ou les arbres vides et notés ■ dans la suite).

La profondeur moyenne des nœuds d'un arbre a est la moyenne des profondeurs de tous les nœuds (internes et externes) de a. Par exemple, la profondeur moyenne des nœuds de l'arbre ci-dessous est $\frac{16}{9}$ car il contient 9 nœuds, respectivement aux profondeurs 0, 1, 1, 2, 2, 2, 2, 3 et 3 (la somme de toutes ces profondeurs est 16).

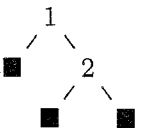


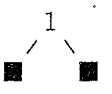
1. Quelle est la profondeur moyenne des nœuds de l'arbre ci-dessous ?



2. Donner un algorithme pour calculer le nombre de nœuds (internes et externes) dans un ABR de racine a : `NbNoeuds(ABR a) : entier`
3. Donner un algorithme pour calculer la somme des profondeurs des nœuds (internes et externes) d'un ABR de racine a : `SommeProf(ABR a) : entier`

   Appliquer votre algorithme sur les trois ABR ci-dessous et donner le résultat renvoyé par `SommeProf` :





1. Le barème est donné à titre indicatif.

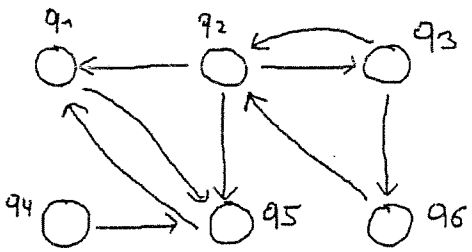
4. Déduire des questions précédentes un algorithme pour calculer la profondeur moyenne des nœuds d'un ABR : `ProfMoyenne(ABR a) : entier`
5. Adapter vos algorithmes pour calculer maintenant la profondeur moyenne des nœuds internes : `ProfMoyenneInterne(ABR a) : entier`
6. Donner un exemple d'ABR ayant une profondeur moyenne inférieure (ou égale) à 3 et avec une hauteur supérieure (ou égale) à 5.
   Rappel : la hauteur d'un arbre est la profondeur maximale de ses nœuds.
   ($\star$) Plus généralement, soit m fixé, quelle forme a un arbre binaire ayant une profondeur moyenne m et une hauteur maximale ? Expliquer.

Exercice 2 : Graphes – (12 points : 1/1/1/2/2/1/2/2)

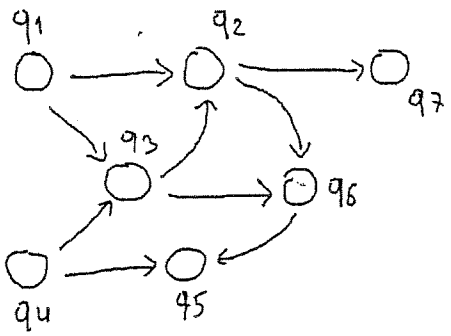
On considère un graphe orienté $G = (S, A)$. On va s'intéresser à l'algorithme de parcours en profondeur (redonné en annexe) et à des variantes de celui-ci.

Toutes les questions qui suivent sont indépendantes...

1. Appliquer l'algorithme de parcours en profondeur sur l'exemple ci-dessous. Donner les dates d[−] et f[−] obtenues, ainsi que la table des prédécesseurs $\Pi[-]$.
   Dessiner la forêt de parcours G_Π contenant les arcs de la table Π obtenue.



2. Donner un graphe $G = (S, A)$ tel que $S = \{q_1, q_2, q_3\}$ et $q_1 \rightarrow_G^* q_2$ et $q_2 \rightarrow_G^* q_1$ et tel qu'il existe un parcours en profondeur sur G pour lequel il n'y a pas de chemin dans G_Π de q_1 à q_2 ni de q_2 à q_1 (c'est-à-dire $q_1 \not\rightarrow_{G_\Pi}^* q_2$ et $q_2 \not\rightarrow_{G_\Pi}^* q_1$).
3. Appliquer un parcours en profondeur sur le graphe ci-dessous : Y a-t-il des arcs retour ? Expliquer.



4. Etant donnés une table des prédecesseurs $\Pi[-]$ renvoyée par l'algorithme de parcours en profondeur pour un graphe $G = (S, A)$ et deux sommets x et z de S, donner un algorithme qui retourne Vrai si et seulement si il existe un chemin de x à z dans la forêt G_Π :

`Chemin`(Π, x, z) : `booléen`

Rappel : la forêt G_Π contient les sommets S et les arcs $(\Pi(x), x)$ pour tout sommet x tel que $\Pi[x] \neq$ nil.

5. Adapter le parcours en profondeur pour obtenir un algorithme qui, étant donné un sommet x d'un graphe $G = (S, A)$, calcule le nombre de sommets atteignables depuis x dans G (rappel : tout sommet est atteigable depuis lui-même).

   `NbSomAtt`(x) : `entier`

6. Adapter le parcours en profondeur pour obtenir un algorithme qui, étant donnés deux sommets x et z, renvoie Vrai si et seulement si il existe un chemin de x à z dans G.

7. Reprendre la question précédente en proposant un algorithme qui s'arrête d'explorer le graphe dès qu'un chemin entre x et z a été trouvé (si votre algorithme de la question précédente est déjà comme cela, vous avez déjà résolu la question !)

8. Montrer que les dates d[−] et f[−] renvoyées par l'algorithme de parcours en profondeur vérifient la propriété suivante pour tous sommets $x, y \in S$:

$$\mathsf{d}[x] < \mathsf{d}[y] < \mathsf{f}[y] < \mathsf{f}[x] \quad \text{ou} \quad \mathsf{d}[x] < \mathsf{f}[x] < \mathsf{d}[y] < \mathsf{f}[y] \quad \text{ou} \quad \mathsf{d}[y] < \mathsf{f}[y] < \mathsf{d}[x] < \mathsf{f}[x]$$

Annexe

On rappelle ici l'algorithme de parcours en profondeur vu en cours pour un graphe $G = (S, A)$: la procédure principale `PP` et la procédure auxiliaire `PP-Visiter`. (Rappel : *temps*, d[−], f[−] et Π sont des variables globales).

Procédure `PP`(G) :
début
  `//Initialisation:`
  pour chaque $x \in S$ **faire**
    Couleur$[x]$:= blanc;
    $\Pi[x]$:= nil;
    d$[x]$:= 0; f$[x]$:= 0;
  temps := 0 ;
  pour chaque $x \in S$ **faire**
    si *Couleur*$[x]$ = *blanc* **alors**
      `PP-Visiter`(G, x) ;
  retourner (d, f, Π)
fin

Procédure `PP-Visiter`(G, s) :
début
  Couleur$[s]$:= gris;
  temps + + ;
  d$[s]$:= *temps*;
  pour chaque $(s, u) \in A$ **faire**
    si *Couleur*$[u]$ = *blanc* **alors**
      $\Pi[u] := s$;
      `PP-Visiter`(G, u) ;
  Couleur$[s]$:= noir ;
  temps + + ;
  f$[s]$:= *temps* ;
fin