UNICAEN

Année 2019-2020

Portail Mathématiques, Informatique
Licence deuxième année (L2 Info, L2 maths, L2 MIASHS)

Algorithmique et structures de données
TD 7 et TD 8

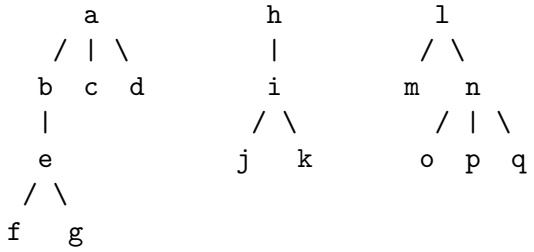
TD 7 – Arbres généraux et forêts

Question 1. Structure de nœud et types forêt et arbreGeneral

Rappelez comment représenter un arbre général ou une forêt sous la forme enfant-fratrie.

Donnez la structure nœud et les types foret et arbreGeneral.

Dessinez la représentation enfant-fratrie de la forêt F_0 ci-dessous.



Question 2. Nombre de nœuds internes

a) Comment définit-on un nœud interne sur F ?

b) Écrivez une procédure *nombreNoeudsInternes(F : foret) : entier* qui renvoie le nombre de nœuds internes de la forêt F.

Question 3. Nombre de nœuds par niveau

Soit N un nœud de profondeur (ou niveau) k d'une forêt F.

a) Quelle est la profondeur de $N \rightarrow premierEnfant$ et $N \rightarrow premierFratrie$?

b) Déduisez-en une procédure *nombreNoeudsNiveau(F : foret, k : entier) : entier* qui renvoie le nombre de nœuds de niveau k.

Question 4. Forêts égales

a) Écrivez une procédure *egalesForets(F, G : foret) : booleen* qui renvoie VRAI si les deux forêts F et G sont égales (mêmes nœuds et même valeurs) et FAUX sinon.

b) On suppose que F possède n nœuds et G m nœuds, avec $n \leq m$.
On définit le coût de la procédure comme étant le nombre de comparaisons de la valeur d'un nœud de F avec celle d'un nœud de G.
Donnez le coût dans le meilleur des cas et dans le pire des cas.
Dans chaque cas, donnez un exemple de forêts F et G.

Question 5. Branches gauche et droite d'un arbre

On définit une branche d'un arbre général comme le chemin (la suite des nœuds) allant de la racine jusqu'à une feuille.

Nous avons donc autant de branches que de feuilles.

a) Expliquez comment parcourir la branche gauche (la branche la plus à gauche).
Donnez la différence avec la branche gauche d'un arbre binaire. Donnez des exemples.
Écrivez une procédure *brancheGauche(F : forêt)* qui affiche cette branche.

b) Expliquez comment parcourir la branche droite (la branche la plus à droite).
Écrivez une procédure *brancheDroite(F : forêt)* qui affiche cette branche.

c) On définit comme coût de ces deux procédures le nombre de nœuds visités.
Comparez le coût des deux procédures.
Pour quelle procédure le coût est lié à la hauteur de la forêt ?

Question 6. * **Degré d'un arbre**

Dans un arbre général, le degré d'un nœud est le nombre d'enfants de ce nœud.
Le degré de l'arbre est le degré maximal de ses nœuds et le degré d'une forêt est le degré maximal de ses arbres.

a) Déterminez le degré de la forêt F_0.
b) Nous sommes sur un nœud N, on définit d comme étant le nombre d'enfants du parent de ce nœud déjà visités (y compris le nœud N).
Comment évolue d sur $N \rightarrow enfant$ et $N \rightarrow fratrie$?
c) Déduisez-en une procédure récursive *degreArbre(A : arbreGeneral, d : entier) : entier* qui renvoie le degré de l'arbre général A. Quelle valeur faut-il mettre à d pour appeler cette procédure ?
d) On souhaite maintenant appeler la procédure sur un arbre différent de None, le test $A = None$ est donc effectué avant l'appel de la procédure.
On nommera *degreArbre2(A : arbreGeneral, d : entier) : entier* cette nouvelle procédure.
Distinguez les quatre cas suivants :
   1. $A \rightarrow premierEnfant = None$ et $A \rightarrow prochainFratrie = None$
   2. $A \rightarrow premierEnfant = None$ et $A \rightarrow prochainFratrie \neq None$
   3. $A \rightarrow premierEnfant \neq None$ et $A \rightarrow prochainFratrie = None$
   4. $A \rightarrow premierEnfant \neq None$ et $A \rightarrow prochainFratrie \neq None$
e) Calculez le nombre d'appels récursifs de *degreArbre2(A : arbreGeneral, d : entier) : entier* en fonction du nombre de nœuds.
Est-ce que ce nombre d'appels dépend de la structure de l'arbre ?

Question 7. * **Parcours en largeur des nœuds d'une forêt**

On considère la procédure *affichageForet* suivante

```
affichageForet(F : foret)
   afficher F->valeur
   affichageForet(F->fratrie)
   affichageForet(F->enfant)
```

a) Dans quel ordre sont affichés les nœuds de F_0 avec la procédure *affichageForet* ?
b) Donnez l'ordre d'affichage des nœuds de F_0 avec le parcours en largeur.
Que faut-il modifier dans la procédure *affichageForet* pour obtenir le parcours en largeur ?
c) Expliquez comment utiliser une file pour effectuer le parcours en largeur.
On ne demandera pas d'écrire une procédure, mais d'exécuter l'algorithme sur F_0.

Pour les plus rapides :

d) Écrivez la procédure *affichageLargeurForet(F : foret)* qui affiche les valeurs des nœuds avec le parcours en largeur.
e) Reprenez la procédure *affichageForet* en échangeant les deux dernières instructions. Étudiez cette nouvelle procédure en l'exécutant sur la forêt F_0.

TD 8 – Arbres binaires de recherche

Question 1. Rappelez la définition d'un ABR. Dans toute la suite, le coût des procédures sera le nombre de comparaisons entre deux entiers.
Donnez la procédure *rechercheABR(A :ABR,x :entier) :ABR* qui retourne un pointeur sur le nœud contenant la valeur x et None si la valeur n'est pas présente.
Soit n le nombre de nœuds de A et h sa hauteur.
Donnez la complexité dans le pire des cas de *rechercheABR* en fonction de h.
Rappelez l'encadrement de h par n. En déduire la complexité dans le pire des cas en fonction de n lorsque A est un arbre complet ?

Question 2. Donnez la procédure *insereFeuille(A :ABR,x :entier) :ABR* qui retourne un ABR enrichi d'un nœud contenant la valeur x inséré au niveau des feuilles.
Comparer la complexité de cette procédure avec celle de la procédure *rechercheABR*.

Question 3. Construisez tous les ABR possibles en insérant des nœuds étiquetés avec les valeurs de l'ensemble $E_3 = \{1, 2, 3\}$. Pour cela, vous utiliserez l'insertion aux feuilles en insérant les valeurs de E_3 dans tous les ordres possibles. Que remarquez-vous ?

Question 4. Écrivez une procédure *afficheCroissant(A :ABR)* qui affiche toutes les valeurs de A par ordre croissant. Comment modifier la procédure pour obtenir l'ordre décroissant ?

Insertion des nœuds à la racine
Dans beaucoup d'applications, les valeurs les plus recherchées sont celles qui ont été insérées récemment. Dans ce cas, il n'est pas judicieux d'insérer les nouveaux nœuds au niveau des feuilles car le coût de la recherche dépend de la profondeur du nœud. Il est préférable d'ajouter le nœud à la racine.

Pour insérer un nœud de valeur x à la racine de A, on sépare A en deux ABR *inf* et *sup* selon la valeur de coupure x de telle sorte que

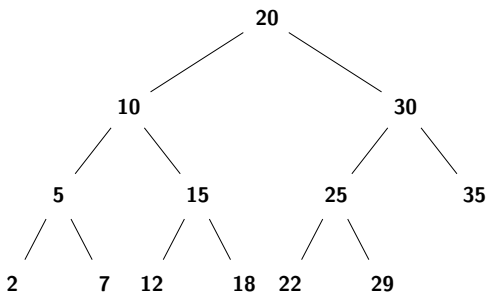
$$\begin{aligned} inf &= \{y \in A : y < x\} \\ sup &= \{y \in A : y > x\} \end{aligned}$$

où $x \in A$ signifie que x est la valeur d'un nœud de A.

Nous souhaitons effectuer cette coupure sans créer de nouveau nœud.

Question 5. Que donne la coupure selon la valeur 20, puis les valeurs 32 et 13 sur l'ABR de la Figure 1.

FIGURE 1 – Exemple d'ABR



Question 6. * Écrivez une procédure récursive *coupeSelon(A :ABR, x :entier) :(ABR,ABR)* qui retourne les deux arbres *inf* et *sup* correspondant à la coupure de A selon la valeur x.

Indication : vous considérerez les quatre cas suivants.

1. $A =$ None
2. x est égal à la valeur de la racine de A
3. x est inférieur à la valeur de la racine de A
4. x est supérieur à la valeur de la racine de A

Montrez comment appeler récursivement la procédure *coupeSelon* dans les cas 3. et 4.

Question 7. Donnez le coût de la procédure *coupeSelon*. Quelle est la complexité dans le pire des cas en fonction de h et en fonction de n ?

Question 8. Écrivez une procédure *insereALaRacine(A :ABR, x :entier) :ABR* qui retourne un ABR enrichi d'un nœud de valeur x inséré à la racine.

Pour cela, vous utiliserez impérativement la procédure *coupeSelon*.

Question 9. Donnez la complexité dans le pire des cas de la procédure *insereALaRacine* en fonction de h et en fonction de n.

Question 10. * Écrivez une procédure récursive *fusionABR(A :ABR,B :ABR) :ABR* qui retourne un ABR C qui est la fusion des ABR A et B. C doit contenir tous les nœuds de A et B, vous ne devez pas créer de nouveaux nœuds. Cette fusion peut être réalisée de plusieurs façons. Nous procéderons de la manière suivante :

1. si $A = None$ alors on retourne B
2. si $B = None$ alors on retourne A
3. si $A \neq None$ et $B \neq None$, on effectue les instructions suivantes
   — on effectue une coupe de B selon la valeur de la racine de A
   — on obtient deux ABR inf et sup
   — on fusionne inf avec $A \rightarrow gauche$ et sup avec $A \rightarrow droit$.

Question 11. * Donnez une majoration du nombre de comparaisons impliquant chaque valeur de A. En déduire une majoration de la complexité dans le pire des cas en fonction de la hauteur h **et** du nombre de nœuds n de A.

On suppose maintenant que les deux ABR A et B sont des arbres binaires complets de hauteur h et que chaque coupe lors de la fusion produit deux ABR de même taille également complets. Soit $C(h)$ le coût de la fusion de deux ABR complets de hauteur h.

Montrez que l'on a la relation

$$C(h) = h + 2\,C(h-1).$$

On montre que l'on a $C(h) \sim 2^{h+1}$.
En déduire le coût de la procédure avec les ABR complets en fonction de n.
Comparez avec la majoration précédente.