TD (4)

AMD5
Partiel du 27 octobre 2014 - durée : 1h30

Les documents et apareils électroniques ne sont pas autorisés.

Exercice 1 :

Écrire une fonction qui, étant donnés un nœud `R` d'un ABR et deux nombres `x` et `y`, renvoie le nombre z de nœuds `N` de l'ABR dont `R` est racine, et tels que $\texttt{x} \leq \texttt{N.val} \leq \texttt{y}$.

Donner sa complexité, exprimée en fonction de la hauteur h de l'ABR et de la valeur retournée z.

Exercice 2 :

1. Écrire une fonction qui renvoie le nombre de nœuds d'un arbre binaire dont la racine est passée en paramètre.
2. Écrire une fonction récursive qui renvoie la somme des profondeurs des nœuds d'un arbre binaire dont la racine est passée en paramètre.
3. En déduire une fonction qui renvoie la profondeur moyenne des nœuds d'un arbre binaire dont la racine est passée en paramètre. Donner sa complexité.
   Note : si la fonction est appelée sur un arbre vide, elle doit afficher un message d'erreur (on ne peut calculer la moyenne d'une distribution vide !).

NB : la racine est à profondeur 0, ses fils à profondeur 1, etc.

Exercice 3 : Union-Find

Le but de cet exercice est d'écrire un algorithme qui affiche le premier ancêtre commun de couples de nœuds d'un arbre.
On suppose que l'on dispose des fonctions suivantes pour manipuler des ensembles de nœuds :

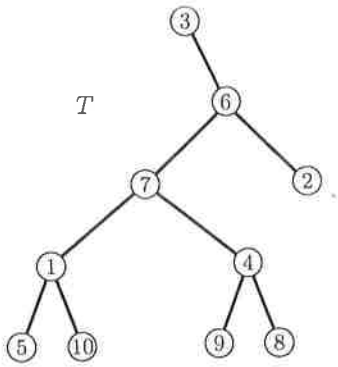
- `CREER(node)` qui prend en paramètre un nœud `node` et crée un ensemble formé de ce nœud qui est donc le représentant de cet ensemble,
- `TROUVER(node)` qui prend en paramètre un nœud `node` et retourne le représentant de l'ensemble auquel appartient le nœud `node` ,
- `UNION(E1,E2)` qui prend en paramètre deux nœuds `E1` et `E2` et fusionne leurs ensemble, de sorte que le représentant l'union est le représentant de l'ensemble de `E1`.

1. Écrire un algorithme qui prend en paramètre la racine `r` d'un arbre binaire (dans lequel on suppose que chaque nœud a des pointeurs vers ses fils et vers son père), et qui parcourt cet arbre en profondeur en effectuant les actions suivantes :
   - un ensemble est créé pour chaque nœud lorsque le nœud est vu pour la première fois dans le parcours ;
   - un nœud est marqué lorsqu'il est vu pour la dernière fois dans le parcours (vous pourrez considérer qu'un nœud a un champ `marque` initialisé à faux) ;

- l'ensemble du nœud node est réuni avec celui de son père en prenant pour représentant le représentant de l'ensemble du père, lorsque node est vu pour la dernière fois dans le parcours.

Donner le nombre d'appels des différentes fonctions CREER, TROUVER et UNION en fonction du nombre n de nœuds de l'arbre.

Exécuter cet algorithme sur l'arbre ci-dessous, en précisant l'évolution des ensembles de nœuds.



2. On souhaite maintenant écrire un algorithme PAC(r,L) qui affiche, pour chaque couple de nœuds d'une liste donnée L de couples de nœuds, leur premier ancêtre commun. On rappelle que le premier ancêtre commun de deux nœuds u et v est l'ancêtre commun à u et v *le plus profond.*

On suppose que tous les nœuds de la liste L sont dans l'arbre. Pour l'arbre T ci-dessus, si L={(5,9),(10,5),(2,1),(4,6),(9,1)}, on veut donc que PAC(3,L) affiche :

```
le premier ancetre commun de 10 et 5 est: 1
le premier ancetre commun de 5 et 9 est: 7
le premier ancetre commun de 9 et 1 est: 7
le premier ancetre commun de 2 et 1 est: 6
le premier ancetre commun de 4 et 6 est: 6
```

Compléter l'algorithme précédent pour qu'il réalise cela (indication : les réponses sont écrites après certains marquages).

Donner le nombre d'appels des différentes fonctions CREER, TROUVER et UNION en fonction du nombre n de nœuds de l'arbre et de la longueur l de la liste L.

DÉF CREER (node)

node {

BTree

Value

fils G

fils D

parent

mark

parcourt Π (r)

if (r != null) {

if (r.filsG == null &&

r.filsG = mark) && (r.filsD == null || r.filsD))

r.mark true;

parcourt Π (r.parent);

else if (r.filsG == null || r.filsG = mark)

if r.filsG == null ||

parcourt Π (r.filsD);

parcourt Π (r.filsD);

else parcourt Π (r.filsG)

si fils G = null et fils D = null CREER(n)

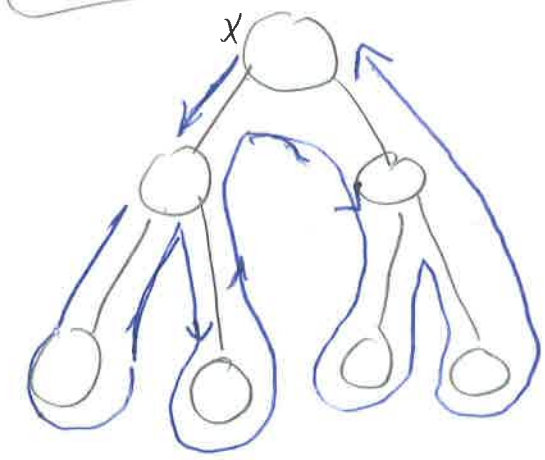
CREER(n)

filsG = mark

il y a au moins un fils et au moins qui n'est pas marqué

parcours profondeur

visiter tous les noeuds d'abord



marqué : passer en dessus