IUT des Pays de l'Adour - Dpt RT - BUT2
R308 - Consolidation de la programmation

Un peu plus loin avec les Structures de Données Dynamiques
Arbres binaires

Manuel Munier

Version 23 septembre 2024 (15:58)

Objectifs

Ce quatrième TP sera pour vous l'occasion de vous d'approfondir la manipulation de structures de données dynamiques (SDD). Nous travaillerons cette fois-ci sur les arbres binaires : chaque nœud (ou cellule) contient 1 valeur et 2 suivants (2 "liens" : 1 vers le "fils gauche" et 1 vers le "fils droit", qui sont tous les deux des arbres). Pour la représentation d'une telle SDD en Python nous réutiliserons le code OO vu à la fin du cours.

Consignes de programmation

Voici quelques conseils pour prendre de bonnes habitudes en programmation avec les SDD :

- Faites des schémas sur papier pour "visualiser" vos SDD.
- Pour les algorithmes, tracer sur papier les différentes étapes avant de coder. Pensez notamment :
  - les références sont à sens unique (on ne peut pas revenir en arrière sans référence explicite)
  - si vous n'avez pas stocké la référence d'un nœud, celui-ci existera toujours en mémoire mais vous ne pourrez plus y accéder !
- Penser à commenter vos codes sources : rôle d'une classe, fonctionnalité des méthodes, etc.
- Penser "modularité"...

Travail à réaliser

Vous allez donc reprendre comme base de travail le code Python vu en cours pour représenter les listes chaînées (classe `Node`, classe `LinkedList`, etc.), en les adaptant ici aux arbres : classe `Node` et classe `Tree`. Pour mémoire, la classe `Tree` (avec 1 attribut `root` référençant le 1er nœud, s'il existe) nous permettra de représenter des arbres vides, i.e. des arbres qui n'ont même pas de premier nœud.

Voici dans les grandes lignes les principaux points à développer dans ce quatrième TP :

1. Écrire la structure de base des classes `Node` et `Tree` avec leurs attributs et leurs constructeurs.
2. Réfléchir (sur papier) puis implémenter (en Python) les méthodes suivantes (méthodes définies sur la classe `Tree`, mais vous aurez "peut-être" besoin de définir d'autres méthodes sur la classe `Node` également...) :
   (a) une méthode `printTree` qui permet d'afficher, de manière récursive, les différentes valeurs de l'arbre ; parcours "en profondeur d'abord"...
   (b) une méthode `countNodes` qui retourne le nombre de nœuds dans l'arbre

(c) une méthode `insert` qui ajoute une valeur "au bon endroit" dans un arbre binaire trié (en créant bien évidemment un nouveau nœud) ; un arbre est dit trié si, quel que soit un nœud de cet arbre, toutes les valeurs de son fils gauche sont plus petites que la valeur portée par ce nœud, et toutes les valeurs de son fils droit sont plus grandes

(d) une méthode `find` qui recherche si une valeur est présente ou pas (`true`/`false`) dans un arbre trié

(e) une méthode `height` qui calcule la hauteur d'un arbre, c'est-à-dire le nombre maximal de "niveaux", quelle que soit la branche ; algorithme récursif également...

3. Plus, bien évidemment, des programmes de test en tout genre pour instancier des objets à partir des classes écrites ci-dessus, invoquer des méthodes sur ces objets, etc.

Document rédigé en LaTeX sous Linux.