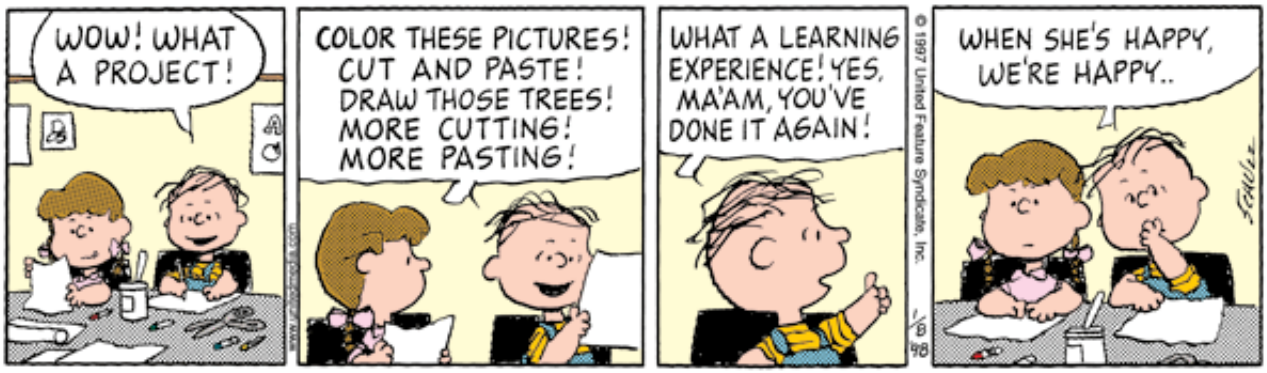
Algorithmique
Partiel n° 2 (P2)

Info-sup (s2)
Epita

9 juin 2015 - 10:00 (308436.75 BW)

Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - — Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - — Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - — Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - — Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Les algorithmes :**
  - — Tout algorithme doit être écrit dans le langage Algo (pas de C, Caml ou autre).
  - — Tout code Algo non indenté ne sera pas corrigé.
  - — Tout ce dont vous avez besoin (types, routines) est indiqué en **annexe** (dernière page) !
- ☐ Durée : 2h00



Exercice 1 (Arbre 234 - Propriétés et insertions - *4 points*)

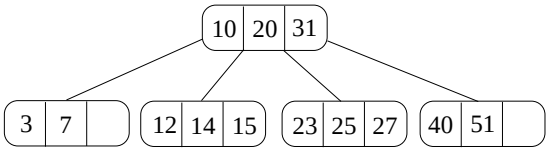


FIGURE 1 – Arbre 2.3.4.

1. Rappeler deux propriétés d'un arbre 2.3.4.
2. Quel problème pose une insertion classique dans un arbre 2.3.4. ?
3. Quelle technique utilise-t-on pour le résoudre ?
4. Insérer successivement les clés 4, 11, 9 et 18 dans l'arbre de la figure 1 en utilisant la méthode "à la remontée". *Dessiner les quatre arbres 2.3.4. (après chaque ajout).*

Exercice 2 (Arbre binaire et code préfixe – *7 points*)

Une méthode pour compresser des fichiers textes consiste à encoder les caractères par des mots binaires. On considère ici un encodage de taille variable : chaque caractère peut être représenté par un nombre différent de bits. Il est évidemment conseillé d'utiliser des codes courts pour les caractères fréquents, et de réserver les codes longs pour les caractères rares.
Il faut d'autre part qu'aucun code ne soit préfixe d'un autre. Par exemple si on choisit d'encoder 'a' par 11 et 'b' par 111, on ne saura plus si 11111 désigne 'ab' ou 'ba'.

1. Considérons le code suivant :

lettre	a	b	c	d	e	f
code	0	101	100	111	1101	1100

   Décoder 110111001100010011101.

2. L'encodage est représenté par un arbre dont les **feuilles** sont les lettres à encoder (le champ `cle` contient la lettre) : le code d'une lettre se lit en suivant la branche de cette lettre à partir de la racine (fils gauche = 0, fils droit = 1).
   À quoi correspond le code d'une lettre ?
3. Dessiner l'arbre correspondant au code de la question 1.
4. L'arbre représentant l'encodage est un arbre localement complet dont toutes les feuilles sont utilisées. Écrire un algorithme qui affiche le code d'une lettre donnée si elle existe dans l'arbre (type donné en annexe). Par exemple, avec le code de la question 1, si la lettre donnée est 'e', l'algorithme affichera "1101".

   Deux versions possibles pour cet algorithme :
   — La "simple" mais pas très optimale : qui parcourt tout l'arbre.
   — La plus optimale (mais...) : qui s'arrête dès que possible. Si cette version est choisie, elle devra de plus afficher "no code found" si la lettre n'est pas trouvée dans l'arbre.

   Dans les deux cas, vous devez écrire un algorithme récursif et son algorithme d'appel.

   Choisissez la version qui vous convient sachant que c'est bien entendu la deuxième version qui rapportera le plus de points. [1]

1. Des fois, il vaut mieux peu de points que pas de point.

Exercice 3 (Tas – *2 points*)

Les deux arbres ci-dessous sont des tas. Les seules opérations autorisées sont l'ajout d'un nouvel élément et la suppression du minimum.

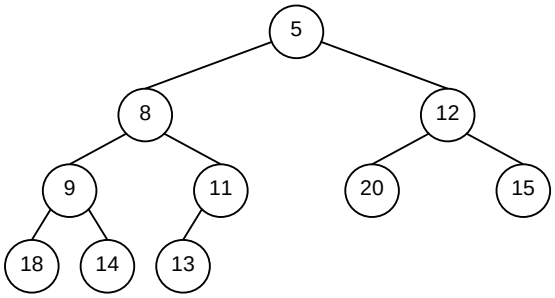


FIGURE 2 – Tas n° 1

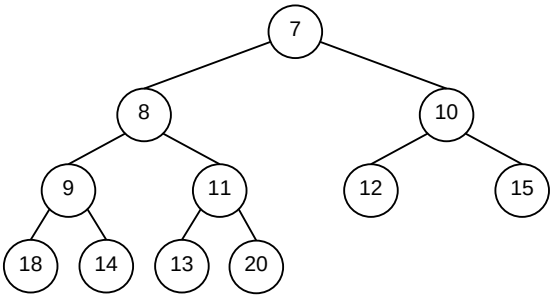


FIGURE 3 – Tas n° 2

Il est possible de passer en 3 opérations du tas n° 1 (figure 2) au tas n° 2 (figure 3).

1. Quelles sont ces 3 opérations ?
2. Dans quel ordre doivent-elles être exécutées ?

Exercice 4 (AVL - Suppression du minimum – *8 points*)

Nous nous intéressons ici à la suppression du minimum dans un AVL avec rééquilibrage.

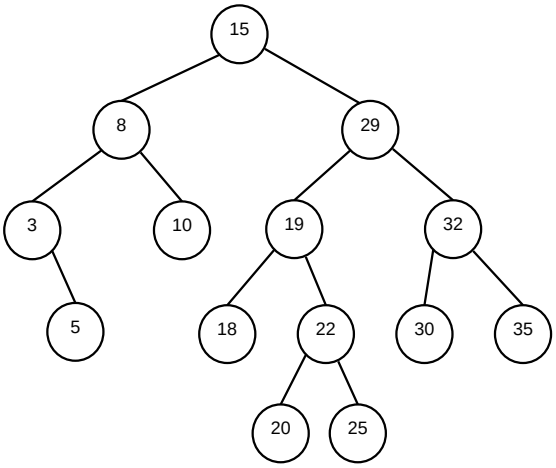


FIGURE 4 – AVL

1. Compléter le tableau donné afin qu'il indique pour chaque cas de déséquilibre (deseq), uniquement après la suppression du minimum, quelle est la rotation à effectuer ainsi que le changement éventuel de hauteur induit ($\Delta h = 0$ si l'arbre ne change pas de hauteur après rotation, 1 sinon).
2. Supprimer le minimum de l'arbre de la figure 4 : donner uniquement l'arbre final (après suppression et éventuelles rotations). Indiquer les éventuelles rotations utilisées (par exemple si une rotation gauche a été effectuée sur l'arbre de racine 42, indiquer *rg(42)*).
3. Écrire la fonction récursive qui supprime la valeur minimum d'un AVL (avec mises à jour et ré-équilibrage à la remontée), donne en résultat la valeur de la clé supprimée, et retourne un booléen indiquant si l'arbre a changé de hauteur. Vous pouvez utiliser les algorithmes `rg, rd, rgd, rdg` qui effectuent les rotations (avec mises à jour des déséquilibres). Le type pour les AVL est rappelé en annexe.

Annexes

Implémentations des arbres binaires

Les arbres contiennent ici des caractères.

```
types
    t_arbreBinaire = ↑ t_noeudBinaire
    t_noeudBinaire = enregistrement
        caractere          cle
        t_arbreBinaire   fg, fd
    fin enregistrement t_noeudBinaire
```

Implémentation des AVL

Le type `t_element` est un type simple muni d'une relation d'ordre.

```
types
    t_avl = ↑ t_avl_node
    t_avl_node = enregistrement
        t_element    cle
        entier        deseq
        t_avl        fg, fd
    fin enregistrement t_avl_node
```

Routines sur AVL

Vous pouvez utiliser toutes les routines implémentant les rotations : `rd, rg, rdg, rgd`. Chacune effectue une rotation sur l'arbre passé en paramètre ainsi que la mise à jour des déséquilibres.

Routines autorisées

Gestion de la mémoire

Il est bien entendu autorisé (souhaité...) d'utiliser les procédures `allouer` et `liberer`.

Divers

Vous pouvez utiliser les fonctions entières `abs, max, min`.

Pour l'affichage, utilisez la procédure `ecrire`

On rappelle que l'opérateur de concaténation des chaînes est : +