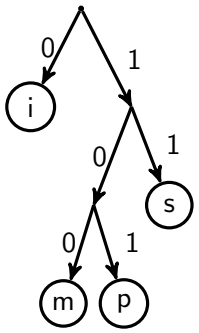
TD 3 - Arbres (applications)

Exercice 1. Algorithme de Huffman



1. Décoder le texte suivant en utilisant le code préfixe décrit par l'arbre ci-contre.
   1 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0
2. Calculer le code de Huffman pour le texte "`abracadabra`", puis donner la séquence binaire correspondante.
   NB : plusieurs solutions sont possibles, mais elles ont toutes la même longueur.

Exercice 2. Arbres binaires de recherche

1. Comment parcourir un ABR pour afficher la liste des éléments en ordre croissant ?
2. Quelle est la complexité de la fonction `find` pour un ABR de n nœuds et de hauteur h ?
3. Écrire une fonction permettant d'insérer un élément dans un arbre binaire.
   `void insert(Arbre* a, Element e);`
4. Calculer l'arbre obtenu en insérant successivement les entiers de 1 à 10 dans l'ordre suivant : 5 8 1 3 4 6 10 7 9 2
5. Écrire des fonctions calculant le minimum et le maximum d'un arbre binaire de recherche. Quelles sont leurs complexités ?
   `Element minBST(Arbre a);`
   `Element maxBST(Arbre a);`
6. Écrire une fonction permettant de tester si un arbre binaire est un arbre binaire de recherche.
   `int isBST(Arbre a);`
   — En utilisant les fonctions `minBST` et `maxBST`. Quelle est sa complexité ?
   — Peut-on améliorer la complexité en utilisant une autre méthode (penser à la question 1) ?
7. Écrire une fonction permettant de supprimer un élément dans un arbre binaire.
   `void delete(Arbre* a, Element e);`

Exercice 3. Arbres AVL

1. Calculer la balance de chaque noeud de l'arbre ci-dessous

Calculer les arbres obtenus successivement après les opérations suivantes :

2. Insertion de 5
3. Insertion de 6
4. Insertion de 4