Travaux Pratiques d'Algorithmique et Programmation n°5

AP3

—Licence Informatique, 3ième année, 1ier semestre—

L'objectif de ce TP est d'apérimenter avec les ABR afin d'étudier leur déséquilibre lorsque l'on construit des ABR à partir de données aléatoire ou ayant une certaine régularité.

Les arbres binaires de recherche (ABR) sont très utiles pour implanter des structures de données pour lesquelles les opérations de recherche, d'insertion et de suppression doivent être rapides (i.e. avoir une bonne complexité en temps).

Les ABR sont généralement assez performants lorsque les données insérées sont aléatoires, mais ils se comportent plutôt mal dans le cas contraire. Même s'il facile de voir qu'en insérant une liste ordonnée d'entiers dans un ABR on obtient un arbre qui à la forme d'un peigne, il vous est proposé d'expérimenter pour vérifier ce point.

1. À partir du module sur les ABR que vous avez réalisé au TP5 et en utilisant la fonction `Random.self_init` du module [1] `Random` qui permet d'initialiser un générateur de nombres aléatoires et la fonction `Random.int borne` qui donne un nombre aléatoire compris entre `0` et `borne - 1` (le paramètre `borne` doit être inférieur à 2^{30}), écrivez une fonction `bst_rnd_create` qui crée un arbre binaire de recherche.
2. Une mesure du déséquilibre d'un arbre est la différence entre la hauteur de son fils gauche et la hauteur de son fils droit. À l'aide de la fonction `bst_rnd_create` estimez le déséquilibre moyen des arbres binaires de recherche construits à partir de suites de nombres entiers aléatoires. Pour avoir des estimations significatives, il est nécessaire de répéter un grand nombre de fois l'estimation de la moyenne faite sur un grand nombre d'arbres.
3. (a) La mesure précédente du déséquilbre d'un arbre binaire ne tient pas compte de cas particuliers qui peuvent rendre inéfficace la recherche d'un élément dans un ABR. Trouvez quelques exemples de ces cas particuliers.

   (b) Une façon plus intéressante de mesurer le déséquilibre d'un arbre binaire est de vérifier, qu'en tout noeud, la hauteur des sous-arbres gauche et droit ne diffère qu'au plus de 1. On dit alors que l'arbre est H-équilibré. Écrivez une fonction `isHbalanced` qui vérifie si un arbre est H-équilibré. Il y a plusieurs manières de faire (plus ou moins efficaces), par exemple, il peut intéressant de « stocker » des informations de déséquilibre aux noeuds de l'arbre

(1). Pour faire des expérimentations, la fonction `Random.init` peut aussi vous être utile. Il vous est conseillé d'aller consulter la documentation du module `Random` dans le manuel d'OCaml disponible sur `ocaml.org`

(c) Adaptez l'expérimentation précédente en considérant les arbre H-équilibrés.

4. Reprenez le même processus d'estimation du déséquilibre moyen d'arbres binaires de recherche, mais cette fois-ci avec des suites de nombres entiers qui contiennent des sous-suites ordonnées de longueurs variables. Les longueurs de ces sous-suites pourront être choisies aléatoirement ou bien de longueur fixe ou encore de longueurs croissantes ou décroissantes.
5. Pour chacune des expérimentations précédentes, établissez un compte-rendu d'expérimentation.