TD Algo2 – session 6 – Extension de structure de données

1er avril 2025

Objectifs d'apprentissage :
— concevoir des algorithmes relatifs aux arbres de rangs ;
— concevoir des algorithmes relatifs aux arbres d'intervalles
— mettre en pratique la technique d'extension de structure de données.

Exercice 1 : Rechercher-Rang-Clé

On considère un arbre de rangs T. Écrire une procédure récursive RECHERCHER-RANG-CLÉ(x, k) qui prend en entrée un nœud x et une clé k, et qui retourne le rang de k dans le sous-arbre de rangs enraciné en x. On supposera que les clés de T sont distinctes et que la clé k est présente dans le sous-arbre enraciné en x.

Exercice 2 : Successeur

Étant donné un élément x dans un arbre de rangs à n nœuds, et un entier naturel i, comment peut-on déterminer le i-ème successeur de x (c'est-à-dire le nœud dont le rang est celui de x plus i) dans l'arbre avec un temps $O(\log n)$? On suppose que l'élément recherché est dans l'arbre.

Exercice 3 : Extrémité minimale

Décrire un algorithme efficace qui, étant donné un intervalle i, retourne l'intervalle recoupant i dont le début est le plus petit possible, ou qui retourne $T.nil$ si un tel intervalle n'existe pas.

Exercice 4 : Temps constant

Montrer comment mettre en œuvre les requêtes MINIMUM, MAXIMUM, SUCCESSEUR et PRÉDÉCESSEUR en temps $\Theta(1)$ dans le cas le plus défavorable. Les performances asymptotiques des autres opérations sur les arbres ne devront pas être affectées. (Conseil : ajouter des pointeurs aux nœuds.)

Exercice 5 : Champ rang

Observons que, chaque fois que le champ taille est utilisé dans RÉCUPÉRER-RANG ou DÉTERMINER-RANG, il ne sert qu'à calculer le rang du nœud dans le sous-arbre issu de ce nœud. Supposons donc que l'on décide de stocker dans chaque nœud son rang dans le sous-arbre dont il est la racine. Montrer comment gérer cette donnée lors de l'insertion. (Ne pas oublier que cette opération peut provoquer des rotations).

Exercice 6 : Rotation gauche

Écrire un pseudo code pour ROTATION-GAUCHE agissant sur les nœuds d'un arbre d'intervalles et capable de mettre à jour les champs max en $\Theta(1)$.

Exercice 7 : Récupérer-rang

Écrire une version non récursive de RÉCUPÉRER-RANG.

Exercice 8 : Tous les intervalles

Étant donné un arbre d'intervalles T et un intervalle i, décrire comment tous les intervalles de T recoupant i peuvent être recensés en $O(\min(n, k \log n))$, où k est le nombre d'intervalles présents dans la liste de sortie. Trouver une solution qui ne modifie pas l'arbre.

Exercice 9 : Rechercher-Intervalle-Exact

Suggérer des modifications aux procédures d'arbre d'intervalles permettant de supporter l'opération RECHERCHER-INTERVALLE-EXACT(T, i), qui retourne un pointeur sur un nœud x de l'arbre d'intervalles T tel que $x.int.début = i.début$ et $i.int.fin = i.fin$, ou qui retourne $T.nil$ si T ne contient pas un tel nœud. Toutes les opérations, y compris RECHERCHER-INTERVALLE-EXACT, devront s'exécuter dans un temps en $O(\log n)$ sur un arbre à n nœuds.

Exercice 10 : Hauteurs noires

On souhaite gérer les hauteurs noires des nœuds d'un arbre rouge-noir en tant que champs des nœuds de l'arbre sans que les performances asymptotiques des opérations d'arbre rouge-noir soient affectées. Comment procéder ?

Exercice 11 : Profondeur

Peut-on utiliser un champ supplémentaire dans les nœuds d'un arbre rouge-noir pour gérer efficacement la profondeur des nœuds. Dire pourquoi.

Exercice 12 : Distance-Min

Montrer comment gérer un ensemble dynamique Q de nombres pouvant supporter l'opération DISTANCE-MIN, qui donne la longueur de la différence entre les deux nombres les plus proches dans Q. Par exemple, si $Q = \{1, 5, 9, 15, 18, 22\}$, alors DISTANCE-MIN($Q$) retourne $18 - 15 = 3$, puisque 15 et 18 sont les nombres les plus proches dans Q. Rendre les opérations INSÉRER, SUPPRIMER, RECHERCHER et DISTANCE-MIN les plus efficaces possibles, et analyser leur temps d'exécution.