STRUCTURES DE DONNÉES / DEUXIÈME ANNÉE / EMD N°3 / 04-05

Problème : *Désordre limité*

On désire ranger des articles réduits à leurs clés (à lire) en mémoire secondaire de la manière suivante :
Le système est initialisé avec un arbre de recherche binaire en mémoire centrale AP composé d'un seul nœud (feuille) ayant comme information le 5-upplet [-1, 0,∞, 0, -1] avec la signification suivante :
Champ1 : Tête de la liste gauche de blocs, -1 si elle vide
Champ2 : Nombre d'articles dans la liste gauche de blocs
Champ3 : Une clé (∞ représente la plus grande clé possible)
Champ4 : Nombre d'articles dans la liste droite de blocs
Champ5 : Tête de la liste droite de blocs, -1 si elle vide
La relation d'ordre est définie sur le troisième champ (Clé).

La première clé est rangée dans le premier bloc d'un fichier LÔF (Liste Non Ordonnée avec format Fixe des articles) de tête LS. Le noeud de l'arbre AP devient (LS, 1,∞, 0, -1).
La seconde clé est rangée dans le même fichier LS et le maillon de l'arbre de recherche binaire devient (LS, 2,∞, 0, -1).
Les (n-1) premières clés sont rangées dans le fichier LS et le noeud de AP est (LS, n-1,∞, 0, -1)
La capacité d'un bloc est b. b <<n. Il est clair qu'à ce moment le fichier est formé d'une liste de blocs.

Quand la n-ième clé se présente pour le rangement, elle est insérée dans le fichier LS, puis la liste des blocs est triée selon la clé avant d'être scindée en deux sous listes de blocs de tailles à peu près égales (en nombre de clés) : LS1 et LS2. Si Nbloc désigne le nombre de blocs de cette liste, alors la première liste LS1 contient les Nbloc/2 premiers blocs, la seconde LS2 les Nbloc/2 derniers blocs. Le nœud avec le contenu [LS1, Nbloc*b, Cle_médiane, n-(Nbloc*b), LS2] est ajouté à l'arbre AP contenant maintenant le nœud racine (LS , n ,∞, 0, -1) et à sa gauche [LS1, Nbloc*b, Cle_médiane, n-(Nbloc*b), LS2]
On prendra comme clé médiane la dernière clé du bloc de rang Nbloc/2

Une façon simple de trier la liste de blocs, consiste à charger ses blocs dans un tableau en mémoire centrale, puis trier le tableau et ensuite remettre les clés ordonnées dans les mêmes blocs de la liste.

Par la suite, toute clé inférieure à Cle_médiane est insérée dans la liste de blocs LS1 et toute clé supérieure sera insérée dans LS2. Les doubles sont écartés.

Quand le nombre de clés dans l'une des deux listes de blocs LS1 ou LS2 atteint n, la liste est de nouveau triée puis scindée en deux et la clé médiane est rajoutée à l'arbre AP.
Etc..
De cette manière, toute liste de blocs contient au plus (n-1) clés. Toute liste est non ordonnée.
L'arbre de recherche binaire joue le rôle d'un index. Globalement, il y a un désordre mais il est limité par l'arbre de recherche binaire.

Exemple : ***La figure montre un exemple avec les clés suivantes insérées :*** **nbIF, Awfl, lUmS, el, aDq, Fx, iY, BlEh, ZBvs, MH, hmx, HpfU, NTTq, Zbl, FreP, kSB, Eflo, elS, EJ, Fhms, bm, NWIU, bxUs, mL, rYO, LCC, mRS, QZ, PDnp, yj, Yj**

n= 15 ; b=4 ; L'arbre de recherche binaire a 2 feuilles. Chaque feuille pointe deux listes de blocs.

Une telle structure permet de faire une partition sur un fichier de données. Chaque élément de la partition est associé à un intervalle. A chaque intervalle est associé une liste de blocs. Dans la figure, on a la partition suivante :]- ∞, HpfU] : Bloc1 et Bloc2 /] HpfU, ZBvs] : Bloc5 et Bloc 8 /] ZBvs, iY] : Bloc3 et Bloc4 /]iY, ∞] : Bloc6 et Bloc7.

On désire écrire les modules de recherche, d'insertion, de parcours et de sauvegarde de l'index

→ Pour la recherche , (i) écrire le module de ***Recherche*** avec les paramètres suivants : (Ap , Val , Trouv , Feuille, Sens, Liste) ;
Ap : l'arbre en mémoire ; **Val** : clé à rechercher ; **Trouv** : existence de la clé dans le fichier ; **Feuille** : le dernier nœud de l'arbre qui pointe la liste de blocs. **Sens** : liste gauche ou liste droite. **Liste** : tête de la liste des blocs. **Adrbloc** : adresse du bloc trouvé ou du dernier bloc si val n'est pas dans le fichier (4 pts)

→ Pour l'insertion d'une valeur donnée, développer les modules suivants :
(ii)***Insérer*** une valeur dans la liste de blocs (4pts)
Si après insertion le nombre d'éléments atteint n,
(iii) ***Trier*** la liste de blocs et ***remettre*** les articles dans les mêmes blocs du fichier (4pts)
(iv) puis ***scinder*** la liste en deux (1pts)
(v) enfin ***Mettre à jour*** l'arbre AP (3pts)

→ Pour le parcours, (vi) donner l'algorithme qui donne ***la partition*** (intervalles et les clés des listes de blocs correspondants) (2pts)

Dire comment sauvegarder et restaurer l'index.
→ (vii) Donner le module de ***sauvegarde*** de l'index (2pts)

Notes

Vous pouvez utiliser les fonctions suivantes pour l'accès et la mise à jour du champ Information d'un nœud.
Info_lg : accès à la tête de liste gauche
Info_ng : accès au nombre d'éléments de la liste gauche
Info_Val : accès à la clé
Info_nd : accès au nombre d'éléments de la liste droite
Info_ld : accès à la tête de liste droite
Et Aff_Info_lg, Aff_Info_ng, Aff_Info_val, , Aff_Info_nd, Aff_Info_ld pour les mises à jours correspondantes.

Notez aussi que 'A' <'B'<....<'Z'<'a'<'b'<....'z'