IUT des Pays de l'Adour - RT2
Informatique - Module IC2 - Algorithmique Avancée

Langage C

Sujet 00b : Recherche, insertion, suppression et fusion dans les tableaux

1 Recherche d'un élément dans un tableau non trié

Dans cette première partie, nous ne faisons aucune hypothèse quant à l'ordre des valeurs dans le tableau (⇒ non trié). Pour rechercher un élément dans un tel tableau, il n'y a que la méthode "brutale", c'est-à-dire parcourir tout le tableau jusqu'à trouver la valeur. Si on arrive à la fin du tableau, c'est qu'elle n'y apparaît pas.

1. Ecrire la fonction `RechElt(N,Tab,Val)` qui retourne le rang de la première occurence de la valeur réelle `Val` dans le tableau `Tab` (contenant `N` valeurs significatives), ou `-1` si `Val` est absente.
2. Ecrire la fonction `RechOcc(N,Tab,ID,Val)` qui retourne le rang de la première occurence de la valeur réelle `Val` dans le tableau `Tab` considéré à partir du rang `ID` (ou `-1` si `Val` est absente).
3. Ecrire la fonction `NbOcc(N,Tab,Val)` qui retourne le nombre d'occurences de la valeur `Val` dans le tableau `Tab`.

2 Recherche d'un élément dans un tableau trié (par dichotomie)

Le principe de cette méthode de recherche d'une valeur `Val` dans un tableau consiste à profiter du fait que celui-ci est déjà trié. Puisque le tableau `Tab` est trié, la comparaison de `Val` avec un élément quelconque de `Tab` permet d'éliminer une partie du tableau où la recherche sera inutile. Par exemple, si nous avons `Val<Tab[10]`, il est inutile de rechercher `Val` parmi les valeurs `Tab[i]` avec `i>10` puisque toutes ces valeurs sont elles-mêmes supérieures à `Tab[10]` (cf. tableau trié). Nous pouvons ensuite continuer la recherche sur la partie restante du tableau.

Tout sous-tableau de recherche sera déterminé par ses indices extrêmes qui permettront de calculer l'indice du milieu (de ce sous-tableau). La comparaison se fera sur la valeur de cet élément médian et permettra de déterminer un nouveau sous-tableau réduit pour la recherche. En prenant le milieu, nous supprimons à chaque itération la moitié des valeurs restant à tester. Si nous avons `N` valeurs dans le tableau initial, il nous suffira de faire plus ou moins $log_2(N)$ tests pour savoir si notre valeur est présente ou non. Par exemple, si `N=1024`, alors `10` tests seulement seront nécessaires (à comparer aux `N` tests de l'exercice précédent avec un tableau non trié si la valeur n'y apparaît pas).

4. Ecrire la fonction `Dichot(N,Tab,Val)` qui retourne le rang de la première occurence de la valeur réelle `Val` dans le tableau `Tab` (contenant `N` valeurs significatives), ou `-1` si `Val` est absente.

3 Insertion d'un élément dans un tableau trié

Nous supposons que le tableau `Tab` est trié par ordre croissant. L'objectif est d'insérer une nouvelle valeur `Val` dans le tableau `Tab` tout en le conservant ordonné. Les probabilités font que cette valeur devra

généralement être insérée au milieu des autres valeurs. Du coup, il faudra procéder à des décalages de manière à "lui faire de la place".

La méthode consiste, en partant du dernier élément du tableau (donc la valeur maximale, puisque le tableau est trié), à comparer `Val` à chaque élément de `Tab`. Dans le cas où `Val` est inférieure à l'élément courant, on décale celui-ci d'une position "vers la droite". Dans le cas contraire, on effectue l'insertion en plaçant `Val` dans la dernière position libérée et on arrête le processus.

5. Ecrire la fonction `Inserer(N,Tab,Val)` qui insère l'élément `Val` à sa bonne place dans le tableau ordonné `Tab` et met à jour `N` (et oui, on a une valeur de plus...).

4 Suppression d'un élément dans un tableau trié

6. Ecrire la fonction `Supprimer(N,Tab,Rang)` qui supprime l'élément présent à la position `Rang`, restructure `Tab` par un décalage vers la gauche et met à jour `N`.

5 Fusion de deux tableaux triés

Maintenant que vous êtes expérimentés dans la manipulation de tableaux, nous allons faire un peu plus compliqué. Dans cet exercice, nous considérons deux tableaux triés `Tab1` et `Tab2` contenant respectivement `N1` et `N2` valeurs réelles. Nous voulons regrouper toutes ces valeurs dans un troisième tableau `Tab` qui devra lui aussi être trié. Nous ne vous donnons aucune indication quant à la méthode à utiliser. C'est à vous de réfléchir et de nous en fournir une (**pensée intelligemment**).

7. Ecrire la fonction `Fusionner(N1,Tab1,N2,Tab2,N,Tab)` qui regroupe toutes les valeurs de `Tab1` et `Tab2` dans le tableau `Tab` et met à jour `N` (normalement $N = N1 + N2 \ldots$).

8. Ecrire la fonction `FusionnerFiltrer(N1,Tab1,N2,Tab2,N,Tab)` qui regroupe toutes les valeurs de `Tab1` et `Tab2` dans le tableau `Tab` en supprimant tous les doublons. Autrement dit, si une valeur apparaît plusieurs fois dans `Tab1` et/ou `Tab2`, elle ne devra apparaître qu'une seule fois dans `Tab`. Cette fonction devra également mettre à jour `N` (normalement $N \leq N1 + N2 \ldots$).

6 Opérations ensemblistes

Si vous en arrivez là avant la fin du TP, voici quelques idées de travail supplémentaires pour vous entraîner plutôt que d'aller surfer sur internet !

Supposons que nous ayons deux tableaux triés `Tab1` et `Tab2` contenant respectivement `N1` et `N2` valeurs réelles. Nous voulons écrire plusieurs fonctions prenant ces deux tableaux en paramètre pour en générer un troisième qui contiendra, selon la fonction, l'union, l'intersection, la différence des deux premiers.

9. $Tab = Tab1 \cup Tab2$ Ecrire la fonction `Union(N1,Tab1,N2,Tab2,N,Tab)` qui calcule dans `Tab` l'union des ensembles de valeurs `Tab1` et `Tab2` (et met à jour `N`...).

10. $Tab = Tab1 \cap Tab2$ Ecrire la fonction `Intersection(N1,Tab1,N2,Tab2,N,Tab)` qui calcule dans `Tab` l'intersection des ensembles de valeurs `Tab1` et `Tab2` (et met à jour `N`...).

11. $Tab = Tab1 - Tab2$ Ecrire la fonction `Difference(N1,Tab1,N2,Tab2,N,Tab)` qui calcule dans `Tab` la différence des ensembles de valeurs `Tab1` et `Tab2` (et met à jour `N`...).