Licence S.T.A. - semestre 2
Informatique - A.P.I.1
TD n°4 recherche dans un tableau

Exercice 1 : *Dérouler les algorithmes de recherche*

Voici deux tableaux t_1 et t_2 indicés de 1 à 11 :

t_1 :	6	2	7	5	3	5	3	6	3	5	3
indices	1	2	3	4	5	6	7	8	9	10	11
t_2 :	2	3	3	3	3	5	5	5	6	6	7

Q 1 . Pour chacun des cas suivants, déroulez l'algorithme demandé pour chercher les valeurs 3 puis 4, en précisant le nombre de comparaisons d'éléments du tableau effectuées.

1. recherche séquentielle dans le tableau t_1.
2. recherche séquentielle dans le tableau t_2.
3. recherche dichotomique dans le tableau t_2.

Q 2 . Peut-on faire une recherche dichotomique dans t_1 ?

Q 3 . Lors d'une recherche séquentielle d'une valeur dans un tableau, est-on sûr en cas de succès d'obtenir la case de plus petit indice contenant cette valeur ? Est-ce vrai pour une recherche dichotomique ?

Exercice 2 : *Évaluation du coût d'une recherche dichotomique infructueuse*

Q 1 . Quel est le nombre maximum de comparaisons d'éléments pour une recherche dichotomique infructueuse dans un tableau de taille N ?

Q 2 . Même question pour le nombre minimum.

Exercice 3 : *Quel sous-programme pour rechercher un élément dans un tableau ?*

Q 1 . Quels peuvent être les résultats souhaités dans la recherche d'un élément dans un tableau ?

Q 2 . Si on ne s'intéresse qu'à la présence ou non de l'élément dans le tableau, quel type de sous-programme écriveriez-vous ?
Écrivez son entête.

Q 3 . Si maintenant on s'intéresse à la présence ou non de l'élément dans le tableau ainsi qu'à l'endroit où il se trouve, quel type de sous-programme écriveriez-vous ?
Écrivez son entête.

Exercice 4 : *Petits soucis d'implantation des sous-programmes de recherche*

En TP vous allez implanter les différents sous-programmes de recherche, 6 en tout, en fonction du type de tableau, trié ou non, du type de recherche, séquentielle ou dichotomique, et des résultats désirés.

Q 1 . Que se passe-t-il si on fait une recherche dichotomique pour un élément qui est strictement supérieur à celui de la dernière case du tableau, plus précisemment quelles sont les deux dernières valeurs pour les indices k et l (voir le cours) ?

Q 2 . Si le type `TABLEAU` a été défini par les déclarations suivantes :

```
const N = 25 ;
type INDICE = 1..N ;
type ELEMENT = ..... ;  // sans importance
type TABLEAU = array[INDICE] of ELEMENT ;
```

et la fonction rechercheDicho

```
function rechercheDicho(const e : ELEMENT ; const t : TABLEAU) : BOOLEAN ;
var k,l,m : INDICE ;
    trouve : BOOLEAN ;
begin
    k := low(t) ; l : high(t) ; trouve := FALSE ;
    ...
    ...
end {rechercheDicho} ;
```

Pourquoi le type des variables `k` et `l` pose un problème ? Comment y remédier (examinez le cas où le type `INDICE` est un intervalle, puis le cas où `INDICE` est un type énuméré par exemple le type `JOUR`) ?

Exercice 5 : *Des manipulations standards sur les tableaux (suite)*

Dans tout cet exercice on suppose que le type `TABLEAU` a été déclaré à l'aide des déclarations suivantes :

```
type INDICE =  ..... ;    // un intervalle d'entiers
     ELEMENT = ........ ; // un type auquel on peut appliquer la relation d'ordre <=
     TABLEAU = array[INDICE] of ELEMENT ;
```

Pour toutes les questions on essaiera de répondre suivant que le tableau est trié ou non et éventuellement en utilisant les deux méthodes : séquentielle et dichotomique.

Q 1 . Réalisez les fonctions suivantes :

sansDoublon	:	TABLEAU	$\longrightarrow$	BOOLÉEN
		t	$\longmapsto$	$VRAI$ si toutes les cases contiennent des éléments différents $FAUX$ sinon
nbOccur	:	TABLEAU $\times$ ELEMENT	$\longrightarrow$	CARDINAL
		t, e	$\longmapsto$	le nombre d'occurence de e dans t
indMax	:	TABLEAU $\times$ ELEMENT	$\longrightarrow$	INDICE
		t, e	$\longmapsto$	le plus grand indice k tel que $t[k] = e$

Q 2 . Écrivez une procédure `afficherIndices` à deux paramètres `t` de type `TABLEAU` et `e` de type `ELEMENT` qui affiche tous les indices `k` tels que `t[k]=e`.

Q 3 . Et si les `INDICE` ne sont pas des entiers, quel est l'indice de la case du milieu de `t[i1..i2]`

Exercice 6 : *Diviser pour régner*

On a vu en cours la méthode de dichotomie pour rechercher un élément dans un tableau trié. On a vu que cette méthode en divisant à chaque fois le tableau en deux (plus ou moins) permettait d'obtenir un algorithme qui, dans le pire des cas est en $\log_2 n$ si n est la taille du tableau.

Pourquoi ne pas diviser le tableau en trois parties à chaque fois !

Q 1 . Écrivez une fonction `rechercheTricho` qui applique ce principe.

Est-elle vraiment plus efficace que la dichotomie ?

Q 2 . Et si on pousse à l'extrême ce raisonnement en divisant le tableau en n parties si il a n cases, qu'obtient-on ?