FACULTÉ DES SCIENCES ET TECHNOLOGIES | Université de Lille

L1 MIASH/SESI/PEIP - 2021–2022

Algorithmes et Programmation

TD : Recherche

Exercice 1 *Recherches séquentielles renvoyant un indice*

Nous avons vu en cours des algorithmes de recherche renvoyant une valeur booléenne. Vous allez dans cet exercice réaliser des fonctions donnant l'indice de l'élément trouvé. Le type des fonctions à réaliser est donc `list(x), int, int, x → int`.

Question 1 Avant de vous lancer dans ces implantations, il faut préciser la spécification de ces fonctions de recherche.

1. Si l'élément recherché est présent plusieurs fois dans la liste, quel indice renvoyer ?
2. Idem si cet élément n'est pas dans la liste ?

Question 2 Réalisez une implantation de la recherche séquentielle dans une liste non triée, puis dans une liste triée, qui donne le plus petit indice d'un élément présent dans la liste.

Question 3 Idem pour le plus grand indice.

Exercice 2 *Point trop n'en faut !*

Voici une implantation en Python erronée de la recherche laborieuse vue en cours.

```
def recherche_laborieuse_erronee(t,a,b,x):
  trouve=False
  for i in range(a,b):
    if t[i]==x:
      trouve=True
    else:
      trouve=False
  return trouve
```

Question 1 Pour chacun des deux appels suivants à cette fonction de recherche, présentez sous forme d'un tableau les valeurs successives que prennent les variables `i` et `trouve`, ainsi que la valeur renvoyée par la fonction.

1. `recherche_laborieuse_erronee([3, 1, 2, 4, 1],0,5,1)` ;
2. `recherche_laborieuse_erronee([3, 1, 2, 4, 5],0,5,1)`.

Question 2 Quelle est l'erreur de programmation manifestement commise ? Proposez une réparation de ce code.

Question 3 En fonction de n la longueur de la liste, combien de comparaisons sont effectuées lors d'une recherche laborieuse erronée ?

Question 4 Proposez un nouveau corps de fonction, qui calcule exactement la même chose, donc produit des résultats erronés, –mais bien plus efficacement– que `recherche_laborieuse_erronee`

Exercice 3 *Pour comprendre la recherche dichotomique*

Soit $t =$ | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | une liste d'entiers.

Question 1 Donnez la suite des tranches de liste parcourue lors de la recherche dichotomique de $x = 11$ dans la liste t. Combien de comparaisons d'éléments de la liste sont-elles effectuées ?

Question 2 Même question avec $x = 12$.

Question 3 Même question avec $x = 10$.

Question 4 Dessinez l'arbre des recherches possibles pour une recherche dichotomique dans une liste de longueur 12.

Exercice 4 *Une petite amélioration*

L'algorithme de recherche dichotomique tel qu'il a été présenté en cours est décit par une boucle tantque dont la seule condition est $d < f$, d et f désignant les indices de début et de fin de la tranche courante de la recherche. Cependant il se peut que l'élément recherché soit trouvé avant que cette condition ne soit plus réalisée.

Proposez une nouvelle version de l'algorithme de recherche dichotomique qui tient compte de cette remarque.

Exercice 5 *Recherche dichotomique renvoyant un indice*

Question 1 Réalisez une implantation de la recherche dichotomique qui renvoie un indice plutôt qu'une valeur booléenne.

Question 2 Dans le cas où l'élément recherché est présent plusieurs fois dans la liste, quel est l'indice qui est renvoyé par votre fonction ?

Exercice 6 *Rechercher l'ensemble des indices*

Étant donnés une structure séquentielle ℓ deux indices a et b, et un élément x, réalisez une recherche de l'ensemble des indices i tels que $\ell[i] = x$ et $a \leq i < b$. Votre fonction renvoie un ensemble d'entiers. Réalisez la pour les différents cas de figures

1. ℓ non nécessairement triée ;
2. ℓ triée, recherche séquentielle ;
3. ℓ triée, recherche dichotomique.

Exercice 7 *Ordre sur les dates*

On suppose dans cet exercice que les dates sont représentées par un dictionnaire qui possède trois clés respectivement nommées `'jour'` `'mois'` et `'annee'`, comme dans l'exercice de la fiche sur les dictionnaire. On suppose de plus que les valeurs associées à ces trois clés sont des entiers. On rappelle que la validité des données a été testé lors de la création des dictionnaires. On rappelle qu'on dispose d'une fonction `cree_date` qui permet de construire des dates.

Question 1 Réalisez une fonction `compare_date` paramétrée par deux dates `d1` et `d2`, qui renvoie un entier valant respectivement `-1`, `0` ou `1` selon que respectivement `d1` est avant `d2`, `d1` égale `d2` , `d1` est après `d2`.

```
>>> compare_date ({'annee': 2016, 'jour': 19, 'mois': 2},{'jour': 1, 'mois': 3, 'annee': 2016})
-1
>>> compare_date ({'annee': 2016, 'jour': 19, 'mois': 2},{'jour': 19, 'mois': 2, 'annee': 2016})
0
```

```
>>> compare_date ({'annee': 2017, 'jour': 19, 'mois': 2},{'jour': 19, 'mois': 2, 'annee': 2016})
1
```

On donne maintenant la variable `etudiants` contenant des quadruplets, `nip` (un entier) , `nom` (une chaine), `prenom` (une chaine), `date_de_naissance` (un dictionnaire comme ci-dessus).

```
>>> etudiants = [ (99998132,"Calbuth","Raymond",{'jour':12,'mois':12,'annee':1987}),
...                (99994451,"Talon","Achille",{'jour':7,'mois':11,'annee':1963}),
...                (99996348,"Calbuth","Monique",{'jour':29,'mois':7,'annee':1987}),
...                (99995433,"Blanc-Sec","Adèle",{'jour':17,'mois':4,'annee':1976}),
...                (99997674,"Brisefer","Benoît",{'jour':15,'mois':12,'annee':1960}),
...                (99998324,"Lagaffe","Gaston",{'jour':28,'mois':2,'annee':1957}) ]
```

Question 2 En utilisant la fonction `recherche_seq` vue en cours sans la modifier, expliquer comment définir une fonction `compare` sur les étudiants afin de savoir s'il existe oui ou non :

— un étudiant né le 29 janvier 1965.
— un étudiant né en 1960.
— un étudiant dont l'anniversaire tombe aujourd'hui.
— un étudiant dont le prénom est `'Timoleon'`
— un étudiant dont le nom commence par `'Cal'`
— un étudiant dont le prénom contient exactement un seul caractère `'a'`.

Pour chaque cas vous donnerez la fonction de comparaison.

Question 3 Pour chaque point précédent dire si la recherche est fructueuse ou non sur l'exemple et combien d'appels à `compare` ont lieu

---