DM 1

Correction DM1

On rappelle l'énoncé

EXERCICE #1 ▶ **À rédiger et à rendre sur Chamilo**

1. Modifier l'algorithme de recherche simple pour qu'il renvoie l'indice de la case où l'élément a été trouvé, -1 si l'élément n'a pas été trouvé. Rédiger la preuve de correction complètement.
2. Modifier l'algorithme de recherche dichotomique de la même façon. Qu'est-ce qui garantit que l'indice trouvé est le plus petit possible ?
3. (Bonus) Fournir un algorithme récursif pour la recherche dichotomique. Expliquer.

Solution rédigée par Laure Gonnord, 28 décembre 2022

1- recherche simple On modifie l'algorithme de recherche simple : on renvoie maintenant un entier, qui est un indice du tableau ou bien -1 si on n'a pas trouvé.

```
int recherche1(int t[N], int el){
  for (int i=0; i<N; i++){
    if (t[i]==el) return i;
  }
  return -1;
}
```

On peut prendre comme invariant : "à la fin de la boucle numéro i (ligne après le return i), on a $0 \leq i < n$ (où n est la longueur de t) et, pour tout $k \leq i$, $t[k] \neq val$". C'est le même que pour l'algorithme non modifié.

— c'est clairement vrai pour la première boucle.

— c'est inductif : si on arrive à la fin d'une boucle supplémentaire, c'est que l'on n'a toujours pas trouvé la valeur qu'on recherche.

Il reste à montrer que cela implique la correction de l'algorithme. En effet :

— Si le flot du programme est arrivé à la ligne "return -1" c'est que la boucle "for" a terminé correctement donc en particulier l'invariant est vrai pour $i = n-1$, et donc pour tout $k \leq n-1$, $t[k] \neq val$, *i.e.* la valeur cherchée n'est pas dans le tableau. La valeur retournée est donc correcte.

— Si le programme a terminé sans que le flot soit arrivé au c'est qu'il existe une valeur de i pour laquelle $t[i] = val$, et à ce moment là on a retourné i, ce qui est ce qu'on veut. De plus, comme la boucle for parcourt le tableau par indices croissants, il s'agit du plus petit indice tel que $t[i] = val$.

2- recherche dichotomique Je donne une version Python de la recherche dichotomique itérative, qui renvoie un indice du tableau.

```
def recherche_dicho2(tab, e): #précondition: tab est trié croissant
    idep, ifin = 0, len(tab)-1
    while ifin >= idep:
        ind_milieu = (idep + ifin) // 2
        if tab[ind_milieu] == e:
            return ind_milieu
        if e < tab[ind_milieu]:
            ifin = ind_milieu -1
        else:
            idep = ind_milieu + 1
    return -1
```

Il est clair que l'indice retourné en cas de succès est un indice qui convient. Néanmoins, en prenant le tableau [2,2,2,4,5,6], la première valeur calculée est `ind_milieu=2`, et il est retourné. Or, l'indice minimal cherché est 0. Le programme ne retourne donc pas l'indice minimal.

On rappelle que pour montrer qu'une proposition "pour tout" est fausse on exhibe un contre exemple

3- version récursive cf code fourni sur la page web.

4- bonus à lire : preuve de correction de la recherche dichotomique Prenons l'invariant de boucle suivant : "soit `e` n'est pas dans `tab`, soit `e` est entre `idep` et `ifin`"

- Par initialisation, l'invariant est vrai avant de rentrer dans la boucle.
- En fin de boucle :
  - soit on a trouvé `e` et on est sorti,
  - soit `e` < `tab`[`ifin` + 1] (première branche du `if`), donc d'après la précondition, si `e` est dans `tab` c'est entre `idep` (qui n'a pas changé) et `ifin` : l'invariant est vérifié
  - soit `e` > `tab`[`idep` − 1] (deuxième branche du `if`), donc d'après la précondition, si `e` est dans `tab` c'est entre `idep` et `ifin` (qui n'a pas changé) : l'invariant est vérifié

Pour la correction (dite "faible"), on suppose que le programme termine. Soit on est sorti par `return ind_milieu` et on a bien une position de `e` dans `tab`. Soit on est sorti par le `return -1` et, d'après l'invariant, `e` n'est pas dans `tab` puisque `idep` > `ifin`.