TD n°12 – Recherche dans un tableau trié

On suppose qu'on dispose d'un tableau d'entiers `tab` trié dans l'ordre croissant. On veut écrire une fonction pour déterminer, de manière efficace, si un élément `x` est présent dans le tableau `tab`.

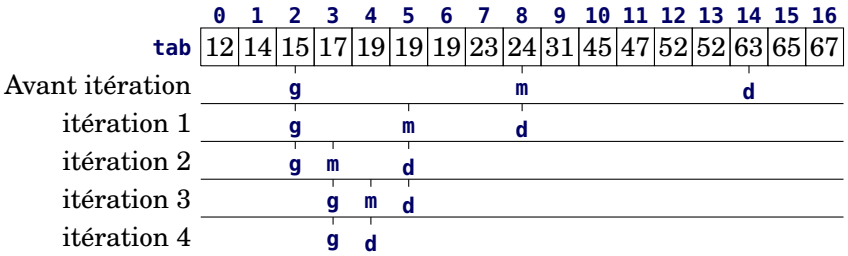
De manière plus précise, on va rechercher la première occurence (si elle existe) de `x` dans `tab` entre les indices `g` exclu et `d` inclus. C'est à dire dans `tab.(g+1)`, `tab.(g+2)`, ... , `tab.(d)`.

Tant que le sous-tableau considéré n'est pas réduit à un élément (c'est à dire tant que `g+1<d`),

- on calcule l'indice `m` du milieu de `g` et `d`. On teste si la valeur `tab.(m)` est supérieure ou égale à la valeur recherchée `x`.
- Si c'est le cas, alors on sait que la première occurence de `x` dans `tab`, si elle existe, sera dans l'intervalle d'indices `]g, m]` donc `d` prend la valeur de `m`.
- Sinon on sait que `x` ne peut apparaître que dans l'intervalle d'indices `]m, d]` et `g` prend la valeur de `m`.

Lorsque le sous-tableau pour les indices dans $]g, d]$ est réduit à 1 élément, alors soit `t.(d)==x` et on renvoie `d`, soit `t.(d)!=x` et on renverra `-1` pour indiquer que `x` n'est pas dans le tableau.

Exemple : recherche de la première occurence de la valeur 19 entre les indices 2 exclu et 14 inclus :



1. Implémenter en Ocaml une fonction `recherche : int array -> int -> int -> int -> int` qui respecte la spécification suivante :
   Entrées : un tableau $t = (t_0, \dots, t_{n-1})$, une valeur x, deux indices g et d.
   Préconditions :
   - t est trié par ordre croissant,
   - $-1 \leqslant g < d \leqslant n-1$.

   Résultat :
   - Si x apparait dans le tableau entre les indices $g+1$ et d, un indice $i \in]g, d]$ tel que $t_i = x$ et pour $g < j < i$, $t_j \neq x$.
   - Sinon -1
2. Quelle valeur donner à g et d pour rechercher dans le tableau entier ?
3. Justifier soigneusement la terminaison de cette fonction.
4. Démontrer que la propriété suivante est un invariant : "Si x apparait dans le tableau, alors il apparait pour la première fois entre les indices $g+1$ et d".
5. Conclure quant à la correction de la fonction `recherche` écrite.

Pour étudier la complexité, on va plutôt regarder la version récursive suivante, qui fonctionne selon le même principe :

```
let rec recherche_rec t x g d =
 if g+1=d then
   if t.(d) = x then d
   else -1
 else let m = (g+d)/2 in
      if t.(m)>=x then recherche_rec t x m d
      else recherche_rec t x g m;;
```

6. De quoi va dépendre la complexité de cette fonction ?
7. On note $n = d - g$ la taille du sous tableau entre les indices `g+1` et `d`. Soit $m = \dfrac{g+d}{2}$ le milieu entre `g` et `d`. Quelle est la taille du sous-tableau d'indices $]g, m]$? Quelle est la taille du sous-tableau d'indices $]m, d]$?
8. En déduire la formule de récurrence de la complexité de la fonction `recherche`.

On suppose que la taille n du tableau initial est une puissance de 2.

9. Trouver la complexité de la fonction sur un tableau de taille $n = 2^k$.