

# TP n°6 - Algorithmes dichotomiques

## 1. Recherche dichotomique

Programmons l'algorithme de recherche dichotomique.

Dans la suite,  $E$  est un ensemble totalement ordonné,  $L$  est un tableau **trié** d'éléments de  $E$  et  $x$  est un élément de  $E$ .

**Q1 :** (*Rappel*) Écrire une fonction `recherche(L,x)` qui opère une recherche de complexité linéaire de  $x$  dans  $L$ .

**Q2 :** *Version récursive de la recherche dichotomique :*

En vous basant sur la description du cours, écrire une fonction `dicho_rec(L,x)` récursive de la recherche dichotomique de  $x$  dans  $L$ .

**Q3 :** *Estimation empirique de la complexité :*

**a)** Importer le module `random` et écrire une fonction `liste_triee(n)` qui renvoie une liste triée de nombres entiers tirés aléatoirement et uniformément entre 0 et 100. *En attendant l'étude des algorithmes de tri, on pourra se servir de la méthode `L.sort()` pour trier la liste  $L$*

**b)** Modifier les fonctions `recherche(L,x)` et `dicho_rec(L,x)` de telle sorte qu'elle renvoie un compteur  $c$  qui compte le nombre de comparaisons effectuées dans chacune des fonctions. On appellera `recherche_complexite(L,x)` et `dicho_rec_complexite(L,x)` ces nouvelles fonctions.

**c)** Comparer les résultats de `recherche_complexite(liste_triee(n),101)` et `dicho_rec_complexite(liste_triee(n),101)` pour  $n=2^k$  où  $k=5,10,15,20,100,\dots$ . Qu'en pensez-vous ?  
Proposer un nom de complexité pour l'algorithme de recherche dichotomique.

**Q4 :** Proposer une version itérative de la recherche dichotomique.

## 2. Exponentiation rapide

Programmons l'algorithme d'exponentiation rapide.

Dans la suite,  $x$  est un nombre et  $n$  est un entier positif. On n'aura bien-sûr jamais recours à l'opérateur `**` dans les questions qui suivent !

**Q1 :** (*Échauffement*) Écrire une fonction `puissance(x,n)` qui renvoie la valeur de  $x$  à la puissance  $n$  de complexité linéaire.

**Q2 :** *Version récursive de l'exponentiation rapide :*

En vous basant sur la description du cours, écrire une fonction `expo_rec(x,n)` récursive de l'exponentiation rapide.

**Q3 :** *Estimation empirique de la complexité :*

**a)** Modifier les fonctions `puissance(x,n)` et `expo_rec(x,n)` de telle sorte qu'elle renvoie un compteur  $c$  qui compte le nombre de multiplication de chaque fonction. On appellera `puissance_complexite(x,n)` et `expo_rec_complexite(x,n)` ces nouvelles fonctions.

**b)** Comparer les résultats de `puissance_complexite(2,n)` et `expo_rec_complexite(2,n)` pour  $n=2^k$  où  $k=5,10,15,20,100,\dots$ . Que pensez-vous de la complexité de l'exponentiation rapide ?

**Q4 :** Proposer une version itérative de l'exponentiation rapide.