TP2 : dichotomie

Exercice 1 : Dichotomie

Q 1.1 Programmer une fonction de recherche

```
function recherche_naive (t : TABLEAU, e : CARDINAL) : CARDINAL;
```

d'un élément dans un tableau dont la complexité est en $\mathcal{O}(n)$ (et pas $\Theta(n)$). La fonction retournera la position de l'élément recherché, zéro si pas trouvé.

Q 1.2 Tester avec la fonction de génération de tableau aléatoire vue dans le premier TP.

Q 1.3 Programmer une fonction de recherche d'un élément dans un tableau *trié*

```
function recherche_trie (t : TABLEAU, e : CARDINAL) : CARDINAL;
```

en utilisant la stratégie par dichotomie.

Q 1.4 Tester sur des tableaux triés grâce à la fonction `tableauCroissant` de l'unité `U_Tableaux`.

Q 1.5 Introduire dans votre programme de test la fonction de tri par insertion vue lors du premier TP et écrire une nouvelle fonction de recherche d'un élément dans un tableau quelconque

```
function recherche (t : TABLEAU, e : CARDINAL) : CARDINAL;
```

et qui utilise la stratégie par dichotomie.

Q 1.6 En utilisant Gnuplot, étudier le comportement asymptotique des fonctions `recherche_naive` et `recherche` :

- tracer la courbe reflétant le comportement de chacune des fonctions pour des tableaux de longueur comprise entre 1000 et 10000 par pas de 500 (100 tirages aléatoires à chaque fois),
- tracer les fonctions permettant de s'approcher au mieux de la courbe expérimentale trouuvée

Exercice 2 : Factures et chèques

Nous allons utiliser les fonctions implantées dans l'exercice précédent afin de résoudre le problème suivant :

> Un artisan dispose de n factures et de p chèques, $p \leq m$. Au dos de chaque chèque est indiqué le numéro de facture correspondant. L'artisan souhaite pointer les factures et les chèques et extraire les factures non payées.

Pour les simulations les factures seront simulées dans un tableau dont les valeurs correspondront au numéro de facture et les chèques dans un second tableau dont les valeurs correspondront au numéro de facture au dos du chèque. Des fonctions pour générer ces tableaux sont fournies dans `U_FacturesCheques`. On les utilisera de la façon suivante :

```
1  program testFacturesCheques;
2
3  uses U_Element, U_Tableaux, U_FacturesCheques;
4
5  var
6     cheq, fact : TABLEAU;
7  begin
8     fact := factures(20);
9     afficheTableau(fact); writeln;
10    cheq := cheques(fact,5);
11    afficheTableau(cheq); writeln;
12 end {testFacturesCheques}.
```

Q 2.1 Proposer une première solution où les factures et les chèques sont pris tel quels. Ecrire une procedure

```
impayes_1(factures, cheques :  TABLEAU)
```

Q 2.2 Quelle est la complexité de cet algorithme ?

Q 2.3 Proposer une seconde solution où les factures OU les chèques, mais pas les deux, sont triés. Ecrire une procedure

```
impayes_2(factures, cheques :  TABLEAU)
```

Q 2.4 Quelle est la complexité de cet algorithme ?

Q 2.5 Proposer une troisième solution où les factures ET les chèques, sont triés. Ecrire une procedure

```
impayes_3(factures, cheques :  TABLEAU)
```

Q 2.6 Quelle est la complexité de cet algorithme ?

Q 2.7 En utilisant Gnuplot, tenter de déterminer pour quel volume de factures et de chèques chacun des algorithmes est le plus intéressant.