Travaux Dirigés d'algorithmique n°8

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Révisions

Cette feuille de TD contient essentiellement des exercices d'examens précédents.

▸ **Exercice 1.** *On considère un tableau* `A` *de* `n` *entiers. Concevoir un algorithme et décrire une fonction C* `Distinct` *qui génère le tableau* `B` *constitué des* `k` *entiers distincts du tableau* `A`*. Analyser la complexité de cet algorithme et en faire la preuve.*
Exemple :
Si `n=5` *et* `A[1]=3`, `A[2]=-10`, `A[3]=3`, `A[4]=5`, `A[5]=5`,
alors `k=3` *et* `B[1]=3`, `B[2]=-10`, `B[3]=5`.
Même question en supposant le tableau `A` *trié.*
(Remarque : si le tableau `A` *est trié, alors le tableau* `B` *doit l'être également. Par contre, si le tableau* `A` *n'est pas trié, alors l'ordre dans lequel apparaissent les éléments* `(3, -10, 5)` *du tableau* `B` *n'a pas d'importance.)*

▸ **Exercice 2.** *Écrire une fonction qui prend en argument une liste chaînée d'entiers et qui la modifie de telle sorte que tous les entiers pairs se retrouvent avant les entiers impairs. Par exemple, la fonction pourra modifier la liste* $(7,4,7,8,4,6,3,9,6)$ *en* $(4,8,4,6,6,7,7,3,9)$ *ou bien encore en* $(8,6,4,4,6,7,3,9,7)$ *: ce qui compte est que les nombres pairs se trouvent finalement avant les impairs.*
Quelle est la complexité de cette fonction ?
Faire la même chose avec un tableau d'entiers plutôt qu'une liste chaînée d'entiers.

▸ **Exercice 3.** *Dans cet exercice, une liste est une liste chaînée de caractères.*
La rotation d'ordre *k d'une liste p est la liste obtenue en ramenant les $k-1$ premiers éléments de p à la fin de la liste p elle-même. Par exemple, la rotation d'ordre 5 de la liste* `(j, o, u, r, b, o, n)` *est la liste* `(b, o, n, j, o, u, r)`.

Écrire une fonction dont le prototype est :

```
Liste Rotation(Liste p, int k);
```

qui retourne la rotation d'ordre k de la liste p si $1 < k \leq Taille(p)$ et retourne la liste p elle-même sinon.

▸ **Exercice 4.** *On considère une liste chaînée $\mathcal{L}$ dont chaque élément est défini par*

```
struct cellule
      {
        int cle;
        struct cellule * suivant;
      };
```

Une variable pointeur `début` *localise le premier élément de $\mathcal{L}$.*
Concevoir et décrire en C une fonction qui élimine dans $\mathcal{L}$ un élément sur deux. Le premier élément de la liste $\mathcal{L}$ est conservé.