Travaux Pratiques d'algorithmique n°5

Cours d'Informatique de Deuxième Année

—Deug SSM M3-MI—

Les structures

Le but de ce TP est de manipuler l'allocation dynamique au travers d'opérations sur les ensembles représentés par une structure à deux champs :

- la taille de l'ensemble ;
- le contenu de l'ensemble, à savoir un tableau d'entiers non trié.

La fin de ce TP sera consacré à un cas particulier : les tableaux contenant les éléments des ensembles sont triés.

---

1 Ensembles

La structure de données pour les ensembles est la suivante :

```
typedef struct {
        unsigned NbElts;
        int *Tab;
        } Ensemble;
```

▸ **Exercice 1. (allocation)** *Ecrire les fonctions* `int AllouerEns(Ensemble *E,unsigned nb)`, `void LibererEns(Ensemble *E)`, `int ReallouerEns(Ensemble *E, unsigned nb)` *et* `int RecopierEns(Ensemble E, Ensemble *Recop)` *pour respectivement allouer de la place pour le tableau d'un ensemble, libérer cet espace alloué, réallouer pour le même ensemble un tableau de taille différente et faire une recopie d'un ensemble dans un autre ensemble.*

▸ **Exercice 2.** *Ecrire les fonctions suivantes :*

- `unsigned MembreEns(int elt, Ensemble E)` *qui teste si l'élément* `elt` *appartient à l'ensemble* `E` *;*
- `void SaisirEns(Ensemble *E)` *qui saisit un ensemble d'entiers ;*
- `void AfficherEns(Ensemble E)` *qui affiche le contenu de l'ensemble* `E`*.*

▸ **Exercice 3. (union, intersection, difference)** *Ecrire les fonctions* `Union`, `Intersection`, `Difference` *et* `DifferenceBis` *qui produisent l'union, l'intersection et la différence de deux ensembles.*
Indication :

$$Difference(E1, E2) = E1 - E2 \ ,$$
$$DifferenceBis(E1, E2) = Union(E1 - E2, E2 - E1) \ .$$

Evaluer la complexité des algorithmes.

▸ **Exercice 4. (inclusion, égalité, complémentaire)** *Ecrire les fonctions suivantes :*

- `unsigned EnsInclus(Ensemble E1,Ensemble E2)` *qui teste si* `E1` *est inclus dans* `E2` *;*
- `unsigned EnsEgaux(Ensemble E1,Ensemble E2)` *qui teste si les deux ensembles égaux ;*
- `unsigned Complementaire(Ensemble E1,Ensemble E2,Ensemble *Result)` *qui retourne le complémentaire de* `E1` *par rapport à* `E2`.

Evaluer la complexité des algorithmes.

2 Opérations sur des tableaux triés

On souhaite maintenant définir quelques opérations sur des tableaux triés. Par conséquent, l'objectif du premier exercice de cette section est la réalisation d'un tri.

▸ **Exercice 5. (tri à bulles)** *Le tri à bulles est un algorithme qui ne nécessite pas de tableau intermédiaire pour réaliser le tri.*
On imagine que le tableau à trier est positionné verticalement. Le but est de "faire remonter à la surface" les éléments les moins lourds. Pour ce faire, on parcourt le tableau de bas en haut (de la fin au début), en échangeant si nécessaire deux éléments consécutifs non triés. A la fin du premier passage, l'élément le plus petit se retrouvera en haut (au début) du tableau. On itère ce processus sur la partie du tableau non triée.

- *écrire l'algorithme du Tri à bulle ;*
- *écrire la fonction* `void TriBulles(Ensemble *E)` *;*
- *évaluer la complexité.*

▸ **Exercice 6. (recherche par dichotomie)** *Ecrire une fonction* `int Recherche(Ensemble E, int x, int g, int d)` *qui retourne* $i + 1$ *si* `x` *est à l'indice* i*, et qui retourne* $-(i + 1)$ *si* `x` *doit être inséré à l'indice* i*.*

▸ **Exercice 7.** *Ecrire la fonction* `int UnionT(Ensemble E1,Ensemble E2,Ensemble *Result)` *dans le cas des ensembles triés.*