Structures de données et algorithmes

Feuille de TD n°2

Recherche d'une valeur majoritaire dans un tableau ou une liste chaînée

Définition liminaire et conventions

On dit qu'une valeur est *absolument majoritaire* dans une suite de n valeurs si elle a plus de $\lfloor n/2 \rfloor$ occurrences dans la suite. Les exercices de cette feuille sont consacrés à diverses façons de rechercher une valeur absolument majoritaire dans un tableau ou une liste simplement chaînée.

Pour les besoins de cette feuille d'exercices, on considère que les listes (simplement chaînées) sont implémentées à l'aide des structures suivantes.

```c
typedef int item;

struct maillon {
  item valeur; /* valeur du maillon */
  struct maillon *suivant; /* adresse du maillon suivant */
};
struct liste {
  struct maillon *premier; /* adresse du premier maillon de la liste */
  int longueur; /* nombre de maillons dans la liste */
};
```

Exercice 1. Recherche quadratique dans un tableau d'entiers (échauffement)

Pour rechercher une éventuelle valeur absolument majoritaire dans un tableau ou une liste, on peut compter le nombre d'occurrences de la première valeur, puis, si nécessaire, le nombre d'occurrences de la deuxième valeur, et ainsi de suite jusqu'à rencontrer une valeur absolument majoritaire ou avoir acquis la certitude qu'il n'en existe pas.

a) Écrivez la définition d'une fonction `maj_tab` qui reçoit en entrées un tableau t et sa longueur n ; elle renvoie un indice (*i.e.* une position) de la valeur absolument majoritaire s'il en existe une, et renvoie -1 sinon. Cette fonction doit implémenter la procédure de recherche esquissée dans le préambule de cet exercice.

b) Écrivez la définition d'une fonction `maj_list` qui reçoit en entrée l'adresse d'une liste ℓ ; elle renvoie l'adresse d'un maillon de la liste dont la valeur est absolument majoritaire, si une telle valeur existe, et renvoie $NULL$ sinon. Cette fonction doit implémenter la procédure de recherche esquissée dans le préambule de cet exercice.

c) Calculez, en fonction de la longueur n du tableau ou de la liste, l'ordre de grandeur, quand $n \longrightarrow \infty$, du nombre de comparaisons de valeurs effectuées lors de l'exécution d'un appel à la fonction de chacune des deux questions précédentes.

Exercice 2. Recherche sous-quadratique dans un tableau d'entiers

On partitionne un tableau de longueur n en deux segments de même longueur, à une unité près : le premier correspond aux positions 0 à $\lfloor n/2 \rfloor - 1$ (segment de longueur $\lfloor n/2 \rfloor$), le second aux positions $\lfloor n/2 \rfloor$ à n (segment de longueur $\lceil n/2 \rceil$).

a) Montrez que si une valeur est absolument majoritaire dans le tableau, alors elle l'est dans au moins un des deux segments définis ci-dessus.

b) Utilisez la question précédente pour écrire la définition d'une fonction **récursive** `maj_tab_bis` qui reçoit en entrées un tableau t et sa longueur n ; elle renvoie un indice (*i.e.* une position) de la valeur absolument majoritaire s'il en existe une, et renvoie -1 sinon.

c) Vérifiez que le nombre de comparaisons de valeurs effectuées lors de l'exécution d'un appel à la fonction de la question précédente est $O(n \log n)$. Pour votre démonstration, vous pourrez supposer que la longueur n du tableau est une puissance de 2.

Exercice 3. Recherche linéaire dans une liste d'entiers

Nouvelle définition. Dans cet exercice, on « durcit » légèrement la définition de la majorité absolue. On dit désormais qu'une valeur est *absolument majoritaire* dans une suite de n valeurs si c'est l'unique valeur de la suite ou si elle a plus de $\lceil n/2 \rceil$ occurrences dans la suite.

On partitionne les maillons d'une liste ℓ en les regroupant par paires de maillons contigus : une paire regroupant les premier et deuxième maillons, une paire regroupant les troisième et quatrième, etc. Il reste éventuellement un singleton, constitué du dernier maillon de la liste si celle-ci est de longueur impaire. On obtient la liste ℓ' à partir de la liste ℓ en comparant les valeurs des maillons à l'intérieur de chaque paire : si elles sont égales, on supprime un des maillons de la paire, sinon on supprime les deux maillons ; on conserve l'éventuel maillon isolé (*i.e.* le singleton).

a) Montrez qu'il existe une valeur absolument majoritaire dans la liste ℓ, elle l'est encore dans la liste ℓ'.

b) Utilisez la question précédente pour écrire la définition d'une fonction `maj_list_bis` qui reçoit en entrée l'adresse d'une liste ; elle renvoie l'adresse d'un maillon de la liste dont la valeur est absolument majoritaire, si une telle valeur existe, et renvoie $NULL$ sinon. **Attention ! Après l'exécution de la fonction, la liste pointée par son paramètre d'entrée contient les mêmes valeurs qu'avant l'exécution**, même si le chaînage peut être modifié. Vous pourrez faire appel aux trois fonctions ci-dessous.

```
1  /* allocation sur le tas pour une liste vide */
2  /* et renvoi de l'adresse de la liste */
3  struct liste *nouvelle_liste();
4
5  /* insertion du maillon d'adresse m dans la liste d'adresse l */
6  void ajouter_maillon_liste(struct liste *l, struct maillon *m);
7
8  /* extraction du premier maillon de la liste d'adresse l */
9  /* et renvoi de l'adresse du maillon extrait */
10 struct maillon *extraire_maillon_liste(struct liste *l);
```

c) Vérifiez que le nombre de comparaisons de valeurs effectuées lors de l'exécution d'un appel à la fonction de la question précédente est $O(n)$. Pour votre démonstration, vous pourrez supposer que la longueur n de la liste est une puissance de 2.