Travaux Dirigés d'informatique n°1

Algorithmique

Rappels sur la complexité et les listes chainées.

Exercice 1 Complexité

On considère la fonction suivante:

```
void F(int n){
int i;
for(i=0;i<n;i++){
   P(n);
   Q(n);
   }
R(n);
}
```

La fonction `P(n)` *s'effectue en temps* $\mathcal{O}(\sqrt{n})$.
La fonction `Q(n)` *s'effectue en temps* $\mathcal{O}(\sqrt{n}\ln n)$.
La fonction `R(n)` *s'effectue en temps* $\mathcal{O}(n)$.
Quelle est la complexité de la fonction `F(n)` *? La réponse devra être justifiée.*

Exercice 2 Complexité.

On considère la fonction suivante:

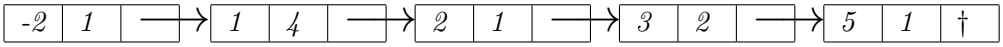
```
void F(int n){
int i;
for(i=0;i*i<n;i++){
   P(n);
   Q(n);
   }
R(n);
}
```

La fonction `P(n)` *s'effectue en temps* $\mathcal{O}(\sqrt{n})$.
La fonction `Q(n)` *s'effectue en temps* $\mathcal{O}(\ln n)$.
La fonction `R(n)` *s'effectue en temps* $\mathcal{O}(\ln n)$.
Quelle est la complexité de la fonction `F(n)` *? La réponse devra être justifiée.*

Exercice 3 Listes triées avec répétitions

On veut manipuler des listes dans lesquelles les valeurs peuvent apparaître plusieurs fois. Pour limiter l'utilisation de place mémoire, on introduit un type `Rliste` *dans lequel chaque cellule contient un champ* `int valeur` *et un champ* `unsigned int repete` *indiquant le nombre de fois où la valeur apparaît dans la liste. De plus les cellules des* `RListe`*s sont triées par valeur croissante.*

Ainsi la liste :



contient :
une fois la valeur -2,
quatre fois la valeur 1,
une fois la valeur 2,
deux fois la valeur 3,
une fois la valeur 5.

1. *Définir le type* `RCellule` *et le type* `RListe` *(pointeur sur* `RCellule`*).*
2. *Écrire une fonction d'allocation d'une* `RCellule` *qui reçoit une valeur n et renvoie, si possible, l'adresse d'une* `RCellule` *indiquant que n apparaît une fois. En cas d'echec la fonction renvoie* `NULL`*.*
3. *Écrire une fonction de recherche d'une valeur dans une* `RListe` *qui renvoie l'adresse de la* `RCellule` *contenant la valeur si elle existe,* `NULL` *sinon.*
4. *Écrire une fonction* `int NombreElement(RListe L)` *qui renvoie le nombre d'éléments mémorisés dans la* `RListe`*, répétitions comprises. Elle renvoie 9 pour la* `RListe` *donnée en exemple.*
5. *Écrire une fonction* `RListe PlusFrequent(RListe L)` *qui renvoie l'adresse de la* `RCellule` *contenant l'élément qui apparaît le plus de fois dans la* `RListe`*.*
6. *Écrire une fonction* `RListe PlusGrand(RListe L)` *qui renvoie l'adresse de la* `RCellule` *contenant du plus grand élément.*
7. *Écrire une fonction d'ajout d'une valeur dans une* `RListe`*.*
8. *Écrire une fonction de suppression d'une occurence d'une valeur dans une* `RListe`*. Si la valeur n'apparait plus dans la liste, la cellule sera supprimée et la place mémoire récupérée.*
9. *Écrire une fonction* `void Fusion( RListe *U, RListe *D)` *qui effectue la fusion des deux RListes dans la première. Les cellules inutiles seront libérées.*