

TD d'algorithmique INF220

TD1 – Récursivité

Le but de ce TD est de manipuler quelques premières fonctions récursives.

Exercice 1

- Q1. Quelle est la valeur renvoyée par **Fibonacci(7)** ? Pour calculer cela, faites la trace du programme en dessinant son arbre d'exécution.
- Q2. Est-ce une méthode efficace en termes de nombre d'additions ? Indiquez sur l'arbre d'exécution, en les entourant d'une même manière, des calculs identiques qui sont faits plusieurs fois.
- Q3. L'algorithme **Mystere** du QCM4 permettait d'obtenir le même résultat pour $n \geq 2$, sans faire plusieurs fois le même calcul. Quel est le nombre d'additions de **Mystere(n)** en fonction de la valeur de n ?
- Q4. Prouvez de manière « visuelle », en utilisant l'arbre de la question Q1, que l'algorithme **Fibonacci** fait toujours plus d'additions, pour toute entrée $n \geq 2$, que l'algorithme **Mystere**.

```
Algorithme Fibonacci
Entrée : entier  $n$ 
Sortie : entier
Variable : entier resultat
Début
  Si  $n=1$  alors :
    resultat  $\leftarrow$  1
  Sinon :
    Si  $n=2$  alors :
      resultat  $\leftarrow$  2
    Sinon :
      resultat  $\leftarrow$  Fibonacci( $n-1$ )+
                        Fibonacci( $n-2$ )
  FinSi
FinSi
renvoyer resultat
Fin
```

```
Algorithme Mystere
Entrée : entier  $n$ 
Sortie : entier
Variables : entiers  $i, j, k$  et  $l$ 
Début
   $i \leftarrow 1$ 
   $j \leftarrow 2$ 
  Pour  $l$  de 3 à  $n$  faire :
     $k \leftarrow i + j$ 
     $i \leftarrow j$ 
     $j \leftarrow k$ 
  Fin Pour
  renvoyer  $j$ 
Fin
```

Exercice 2 – Puissances de 2

- Q1. Écrivez une fonction récursive **deuxPuissance** qui prend en entrée un entier n et renvoie la valeur de 2^n , sans utiliser de boucle (ni **Pour**, ni **Tant que**). *Indications : quel est le cas de base (initialisation) ? Quelle est la formule d'hérédité ?*
- Q2. Vous allez écrire à la question Q6 une fonction récursive **termineTableauPuissancesDeDeux** qui prend en entrée un entier n , et un entier i compris entre 1 et n , et renvoie un tableau d'entiers de n cases, dont chaque jusqu'à la i -ième incluse contient la puissance de 2 correspondante (la première case contient 2^1 , la deuxième $2^2=4$, la troisième $2^3=8$, etc). Que renvoie **termineTableauPuissancesDeDeux(5,4)** ?
- Q3. Écrivez la fonction récursive **termineTableauPuissancesDeDeux** qui répond à la question précédente sans utiliser de boucle (ni **Pour**, ni **Tant que**).
- Q4. Écrivez la fonction récursive **remplitTableauPuissanceDeDeux** qui prend en entrée un entier n et renvoie un tableau d'entiers de n cases dont la i -ième case contient 2^i (appelez la fonction **termineTableauPuissancesDeDeux**).

TD2&3 – Tris

Exercice 1 – Implémentation du tri à bulles

- Q1. La fonction **etapeTriBulles** fait une étape du tri à bulles. Comme nous l'avons vu en cours, elle parcourt le tableau d'entiers fourni en entrée en vérifiant pour chaque case si elle est bien inférieure ou égale à la suivante, et échange leurs valeurs si ce n'est pas le cas. Elle renvoie le tableau ainsi obtenu. Que renvoie **etapeTriBulles**({4,5,1,2,2,9,8}) ?
- Q2. Écrivez la fonction **etapeTriBulles**.
- Q3. Pour savoir si on arrête l'algorithme à la fin d'une étape du tri à bulles, il faut savoir si le tableau est trié. Pour cela, écrivez un algorithme récursif **testTrieAvantCase**, qui prend en entrée un tableau d'entier tab et un entier i , et renvoie VRAI si tab est trié jusqu'à la i -ième case incluse, et renvoie FAUX sinon.
Indication pour l'initialisation : que renvoie **testTrieAvantCase** sur n'importe quel tableau tab en entrée si l'entrée i vaut 1 ?
Indication pour l'hérédité : si le tableau tab en entrée de **testTrieAvantCase** est trié jusqu'à la $(i-1)$ -ième case, comment savoir s'il est trié jusqu'à la i -ième ? Et s'il n'est pas trié jusqu'à la i -ième case ?
- Q4. En utilisant **etapeTriBulles** et **testTrieAvantCase**, écrivez l'algorithme **triBulles** qui prend en entrée un tableau d'entiers et renvoie ce tableau trié par le tri à bulles.

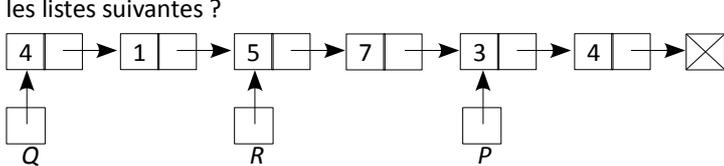
Exercice 2 – Tri par insertion

- Q1. Écrivez un algorithme **decaleApresCase** qui prend en entrée un tableau d'entiers tab et un numéro de case i et décale d'une case vers la droite toutes les valeurs situées après la i -ième case (non incluse). Que renvoie **decaleApresCase**({1,4,5,,,,},1) ?
- Q2. Écrivez un algorithme **trouvePosition** qui prend en entrée un tableau d'entiers trié tab (dont les dernières cases sont éventuellement vides) et un entier k , et qui renvoie le numéro de case de tab après laquelle il faut insérer k pour que tab reste trié. Par exemple, **trouvePosition**({1,4,5,,,,},2) renvoie 1 car il faudrait insérer l'entrée $k=2$ après la première case pour obtenir le tableau trié {1,2,4,5,,,,}.
- Q3. En utilisant les algorithmes **decaleApresCase** et **trouvePosition**, écrivez un algorithme **insereDansTableauTrie** qui prend en entrée un tableau d'entiers trié tab et un entier k , et qui insère k dans tab à la bonne position pour que tab reste trié, et renvoie tab . Que renvoie **insereDansTableauTrie**({1,2,4,5,,,,},2) ?
- Q4. En utilisant l'algorithme **insereDansTableauTrie**, écrivez l'algorithme **triParInsertion** qui prend en entrée un tableau d'entiers tab , crée un tableau vide $tab2$, et le remplit progressivement avec les éléments de tab de telle sorte que $tab2$ reste toujours trié. Finalement, **triParInsertion** renvoie le tableau $tab2$ qui est trié et a été entièrement rempli par les éléments de tab .

TD4 – Listes

Exercice 1 – Manipulation des fonctions de base sur les listes

Q1. Dessinez l'ensemble des listes P , Q et R après chaque instruction de l'algorithme **initialiseListes** ci-contre. Voici par exemple le dessin de la situation après la ligne 1 :



Algorithme initialiseListes

Variables : P, Q, R , listes d'entiers

Début

1. $Q \leftarrow \text{creerListe}(1, \text{creerListe}(2, \text{NULL}))$
 2. $P \leftarrow \text{creerListe}(1, \text{creerListe}(2, \text{creerListe}(3, \text{NULL})))$
 3. $Q \leftarrow \text{creerListe}(4, P)$
 4. $\text{tete}(\text{suivant}(P)) \leftarrow 6$
 5. $R \leftarrow \text{creerListe}(5, \text{suivant}(\text{suivant}(P)))$
 6. $\text{suivant}(P) \leftarrow R$
 7. $P \leftarrow \text{suivant}(R)$
 8. $\text{suivant}(P) \leftarrow \text{creerListe}(\text{tete}(Q), \text{NULL})$
- Fin

Exercice 2 – Écriture de fonctions sur les listes et implémentation d'une file

- Q1. En utilisant une boucle **Tant que**, écrivez en pseudo-code un algorithme **afficheListe** qui prend en entrée une liste d'entiers et affiche le contenu de chaque case. Par exemple, **afficheListe(creerListe(1, creerListe(2, NULL))** affichera 1 puis 2.
- Q2. Sans utiliser de boucle **Tant que**, écrivez en pseudo-code un algorithme récursif **afficheListe** qui prend en entrée une liste d'entiers et affiche le contenu de chaque case. Vous constatez qu'un parcours de liste peut s'effectuer de manière récursive, ou avec une boucle.
- Q3. Écrivez un algorithme **longueurListe** qui prend en entrée une liste L d'entiers et renvoie le nombre de cases de L . Si l'on exécute cette instruction juste après l'instruction 8 de l'algorithme **initialiseListes** de l'exercice 1, que renvoie **longueurListe(R)** ?
- Q4. Écrivez un algorithme **insereAvantCase** qui prend en entrée une liste L d'entiers et deux entiers i et x , et insère avant la i -ième case de la liste une case contenant l'entier x . La question Q2 de l'exercice 1 pourra vous être utile.
- Q5. En utilisant les algorithmes **longueurListe** et **insereAvantCase**, écrivez un algorithme **enfile** qui prend en entrée une liste L d'entiers et un entier a , et insère à la fin de L une nouvelle case contenant a .
- Q6. Écrivez un algorithme **defile** qui renvoie la valeur de la première case de la liste d'entiers en entrée, et supprime cette case.

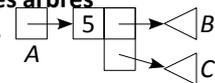
TD5 – Arbres

Exercice 1 – Codage d'un arbre

Que renvoie **écritureParenthesee(T)**, si T est l'arbre représenté à droite ?

Exercice 2 – Comptages dans les arbres

- Q1. On va compter les feuilles d'un arbre d'entiers. Pour cela on définit l'algorithme récursif **nombreFeuilles** qui prend en entrée un arbre d'entiers A , et renvoie le nombre de feuilles de l'arbre. Si l'arbre A a deux fils qui sont les arbres B et C , que **nombreFeuilles(B) = 4** et que **nombreFeuilles(C) = 6**, quelle valeur renvoie **nombreFeuilles(A)** ?



Algorithme écritureParenthesee

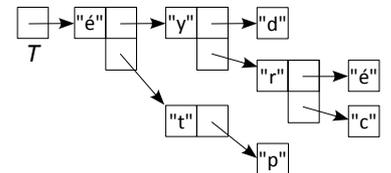
Entrée : arbre de chaînes de caractères A

Type de sortie : chaîne de caractères

Variables : chaîne de caractères $resultat$, entier i

Début

1. Si **estFeuille(A)** alors :
 2. $resultat \leftarrow \text{racine}(A)$
 4. Sinon :
 5. $resultat \leftarrow "("$
 6. Pour i de 1 à **longueur(enfants(A))** faire :
 7. $resultat \leftarrow \text{Concatene}(resultat, \text{Concatene}(\text{écritureParenthesee}(\text{Case}(\text{enfants}(A), i)), ","))$
 8. Fin pour
 9. $resultat \leftarrow \text{Concatene}(\text{Concatene}(resultat, ","), \text{racine}(A))$
 10. Fin si
 11. renvoyer $resultat$
- Fin



Quelle est l'initialisation de l'algorithme **nombreFeuilles** ?

- Q2. Déduisez-en l'écriture de l'algorithme récursif **nombreFeuilles**.
- Q3. En faisant de légères modifications de l'algorithme **nombreFeuilles**, écrivez un algorithme récursif **nombreNoeuds** qui calcule le nombre de nœuds de l'arbre d'entiers fourni en entrée (nœuds = toutes les cases de l'arbre qui contiennent un entier).
- Q4. En appelant les algorithmes **nombreFeuilles** et **nombreNoeuds**, écrivez un algorithme **nombreNoeudsInternes** qui renvoie le nombre de nœuds qui ne sont pas des feuilles dans l'arbre en entrée.

Exercice 3 – Recherche dans un arbre

- Q1. Écrivez un algorithme récursif **recherche** qui prend en entrée un entier i et un arbre d'entiers A , et renvoie VRAI si i se trouve dans une des cases de A , et FAUX sinon.
- Q2. Écrivez un algorithme récursif **profondeur** qui prend en entrée un entier i et un arbre d'entiers A , et renvoie la profondeur de l'entier i (0 s'il est à la racine, 1 s'il est dans un fils de la racine, 2 s'il est dans un de leurs fils, etc.) s'il se trouve dans l'arbre, et -1 sinon.