ESP de Nouakchott
TD et TP d'algorithmique

4 novembre 2014

1 Formalisme d'un algorithme, algorithmes itératifs, calcul de complexité

1. Combien d'instructions effectue l'algorithme ci-dessous ?

   Algorithme 1: Algo1()

   $a \leftarrow 0$
   $b \leftarrow 0$
   afficher(a)
   afficher(b)

2. Calculer le nombre d'instructions dans le cas de N = 5 ? Donner la complexité en fonction de N ?

   Algorithme 2: Algo2(N)

   $res, i : Entier$
   $res \leftarrow 0$
   <u>Pour</u> $i \leftarrow 1$ à N <u>Faire</u>
       $res \leftarrow res + i$
   afficher(res)

3. Calculer le nombre d'instructions dans le cas de N = 5 ? Donner la complexité en fonction de N ? Que fait cet algorithme ?

   Algorithme 3: Algo3(N)

   $i : Entier$
   <u>Pour</u> $i \leftarrow 1$ à N <u>Faire</u>
       $Algo2(N)$

4. Exprimer le nombre d'instruction effectué en fonction de N ?

   Algorithme 4: Algo3(N)

   $i, j, k : Entier$
   <u>Pour</u> $i \leftarrow 1$ à N <u>Faire</u>
       <u>Pour</u> $j \leftarrow i$ à N <u>Faire</u>
           <u>Pour</u> $k \leftarrow j$ à $J + 1$ <u>Faire</u>
               afficher(2)

5. Comparaison de deux tableaux : Écrire un algorithme qui compare deux tableaux d'entier $T1$ et $T2$ de taille respectivement n_1 etn_2 .
   - Si $n1 < n_2$, on retourne -1
   - Si $n1 > n_2$, on retourne 1
   - Si $n_1 = n_2$ on compare les valeurs et on retourne 0 si les deux tableaux contiennent les même valeurs, sinon on considère l'indice k le plus petit indice tel que $T_1[k] \neq T_2[k]$. Si $T_1[k] < T_2[k]$, on retourne -1 sinon on retourne 1

   Étudier sa complexité en meilleur et en pire de cas

2 Récursivité

Écrire des algorithmes récursifs pour calculer les fonctions mathématiques suivantes

1. Calcul de r^n où r est un réel et n est un entier
2. Calcul du terme f_n de la suite de Fibonacci

3 Autres

1. Question d'examen (Devoir 2013-2014)
   On considère l'algorithme suivant :

   Algorithme 5: Algo1(x, y, z : Entier)

   ```
   Entrée : x, y
   PréC : Rien
   Sortie : z
   PostC : {?}
   z ← 0
   I ← 0
   s1 ← 1
   s2 ← 1
   Si x < 0 Alors
       s1 ← −1
   Si y < 0 Alors
       s2 ← −1
   Tant que (I < s2 × y) faire
       z ← z + (s1 × s2) × x
       I ← I + 1
   ```

   (a) Quelle est la valeur retournée par l'algorithme (la valeur de z) dans les cas suivants :

   i. x = 0, y= 0

   ii. x = 0, y =2

   iii. x = 2, y = 0

   iv. x = 2, y= 2

   v. x = -2, y= 2

   vi. x = -2, y= -2

   vii. x = 2, y= -2

   (b) Combien d'opérations de comparaison effectuées par cette algorithme en fonction des données d'entrée ?

   (c) Que fait cet algorithme ? En déduire la post-condition (*PostC*) ?

   (d) Écrire un algorithme récursif équivalant à cet algorithme ?

2. Question d'examen (Devoir 2013-2014)
   On considère l'algorithme suivant :

Algorithme 6: Algo1(T, x , B)

Entrée : $T[1..n]$ tableau de n Entier , x : Entier
PréC : $longueur(T) > 0$
Sortie : B : booléen
PostC : {?}
$I \leftarrow 1$
$j \leftarrow 5$
$B \leftarrow Faux$
Tant que $(I \leq longueur(T) \wedge (\neg B))$ faire
  Si $(T[I] = x)$ Alors
    $j \leftarrow j + 1$
  Si $(j = 7)$ Alors
    $B \leftarrow vrai$
  $I \leftarrow I + 1$

(a) Quelle est la valeur retournée par l'algorithme (la valeur de B) dans les cas suivants (donner la valeur de I et j à la fin de l'algorithme) :
   i. T= [1,2,3,4,5,6] , x= 5
   ii. T= [10,3,3,4,4,6] , x= 4
   iii. T= [5,3,3,4,4,6] , x= 3
   iv. T= [5,3,3,4,4,6] , x= 0

(b) Dans le pire de cas, combien d'opérations de comparaison effectuées par cette algorithme en fonction de $n = longueur(T)$?

(c) Dans le meilleur de cas, combien d'opérations élémentaires effectuées par cette algorithme ?

(d) Que fait cet algorithme ? En déduire la post-condition (*PostC*) ?

(e) Réécrire cet algorithme en utilisant une boucle "Pour" au lieu de "Tant que" ?

4 TP

Implémenter en C, les algorithmes 4 et 5 dans la première section, puis les algorithmes dans les sections 2 et 3.

5 Solution

Algorithme 7: Add($L1$, $L2, L3$: Liste)

```
Entrée : L1, L2
PréC : {}
Sortie : L3
PostC : {L3 = L1(1)L2(1)L1(2)...}
Si (L1 = NULL ∧ L2 = NULL) Alors
   L3 ← NULL
Sinon Si (L1 ≠ NULL ∧ L2 = NULL) Alors
   L3 ← L1
Sinon Si (L2 ≠ NULL ∧ L1 = NULL) Alors
   L3 ← L2
Sinon
   Q ← Allouer(Q)
   L3 ← L1
   Q ← L1
   pos ← 1
   Tant que (L1 ≠ NULL ∧ L2 ≠ NULL) Faire
      Si pos = 1 Alors
         L1 ← L1− > suiv
         (Q− > suiv) ← L2
         pos ← 2
      Sinon
         L2 ← (L2− > suiv)
         (Q− > suiv) ← L1
         pos ← 1
      Q ← (Q− > suiv)
```