Licence de Mathématiques et Informatique
Semestre 2

Université de Perpignan Via Domitia

Algorithmique

Correction de l'examen de seconde session 2015/2016

Durée : 2 heures. Aucun document autorisé.

Exercice 1. (16 points) **/16**

On considère `T` un tableau d'entiers de signe quelconque et de longueur arbitraire n. Si besoin, on peut supposer par exemple que $n \leq 100$. **On supposera que ce tableau est intialisé dès sa déclaration, par exemple avec une fonction `init()`** : `T : tableau[n] d'entiers = init()`

1. Écrire un algorithme qui identifie si le tableau `T` contient ou non, au moins une valeur strictement positive. La valeur ainsi calculée sera affichée en fin d'algorithme.

   Cet algorithme sera écrit sans aucune boucle pour. **/4**

```
declare
  n : constante entier = 100
  T : tableau[n] d'entiers = init()
  i, indice : entier
  res : booleen = VRAI
debut
  i = 0
  tantque (i<n) and (T[i] <= 0) faire
    i = i+1
  fin tantque
  si (i == n) alors # on a parcouru tout le tableau sans succes
    res = FAUX
  finsi
  afficher(``le tableau contient au moins une valeur strictement positive : '', res)
fin
```

2. ($\star$) Quel est le nombre maximal de comparaisons effectuées par cet algorithme ? Ce nombre sera exprimé en fonction de n. **/2**

```
La boucle tantque effectue au plus 2n + 1 comparaisons
Le test si effectue exactement 1 comparaison.
Le nombre de comparaisons est donc : 2n + 2
```

3. ($\star$) Décrire une instance de ce problème qui atteint cette valeur maximale. **/1**

```
Le tableau T qui contient n valeurs négatives ou nulles.
```

4. ($\star$) Quelle est la complexité (en temps) asymptotique de cet algorithme ? **/1**

```
La complexité asymptotique en temps est linéaire en la taille du tableau, ici n.
```

5. Écrire l'en-tête d'une fonction `f` qui effectue le traitement la question 1 sans l'affichage final. **/1**

```
fonction f(n: entier, t[n]: tableau d'entiers) retourne booleen
```

6. Réécrire le traitement de la question 1 à l'aide de la fonction `f`. **/1**

```
declare
  n : constante entier = 100
  T : tableau[n] d'entiers = init()
avec fonction f(n: entier, t[n]: tableau d'entiers) retourne booleen
debut
  afficher(``l'indice cherché vaut : '', f(n, T))
fin
```

7. Écrire le corps de la fonction `f`. **/1**

```
fonction f(n: entier, t[n]: tableau d'entiers) retourne booleen
  i : entier
  rep : booleen = VRAI
debut
  i = 0
  tantque (i<n) and (T[i] <= 0) faire
    i = i+1
  fin tantque
  si (i == n) alors # on a parcouru tout le tableau sans succes
    rep = FAUX
  finsi
  retourner rep
fin
```

8. On va modifier la fonction `f` en une fonction `f2` qui effectue un traitement similaire mais qui retourne l'indice de la première valeur strictement positive présente dans `T` ou -1 si aucune valeur strictement positive apparaît dans le tableau. Écrire l'en-tête de cette fonction `f2`. On ne demande pas le corps de cette fonction. **/1**

```
fonction f2(n: entier, t[n]: tableau d'entiers) retourne entier
```

9. **La fonction `f2` peut être utilisée dans la suite.**
Écrire un algorithme qui calcule la valeur minimale strictement positive d'un tableau `T` d'entiers de longueur arbitraire $n \leq 100$. L'algorithme retournera cette valeur ou 0 si aucune valeur strictement positive n'est présente dans le tableau. La valeur ainsi calculée sera affichée en fin d'algorithme. **/1**

```
declare
  n : constante entier = 100
  t : tableau[n] d'entiers
  indice, min : entier
avec fonction f2(n: entier, t[n]: tableau d'entiers) retourne entier
debut
   indice = f2(n, T)
   si indice == -1 alors
     min = 0
   sinon
     min = t[indice]
     pour i de indice+1 à n-1 faire
       si (t[i] < min) and (t[i]>0) alors
         min = t[i]
       finsi
     fin pour
   finsi
  afficher(``le min > 0 du tableau vaut: '', min)
fin
```

10. ($\star$) Quel est le nombre maximal de comparaisons effectuées par cet algorithme ? Ce nombre sera exprimé en fonction de n. **/1**

```
La fonction f2 effectue au plus 2n + 2 comparaisons.
Le test si effectue exactement 1 comparaison.
La boucle pour effectue au plus 2n comparaisons : à chaque itération, 1 dans
l'itérateur de boucle et une dans le si sut t[i].
Le nombre de comparaisons est donc : 4n + 3
```

11. Décrire une instance de ce problème qui atteint cette valeur maximale. **/1**

```
Le tableau T qui contient $n$ valeurs négatives ou nulles.
```

12. Quelle est la complexité (en temps) asymptotique de cet algorithme ?. **/1**

```
La complexité asymptotique en temps est encore linéaire en la taille du tableau, ici $n$.
```

Exercice 2. (12 points) **/12**

On dispose d'une fonction `min(a,b)` qui retourne le minimum des deux entiers de signe quelconque `a` et `b`. Soit $t = [a_0, a_1, \cdots, a_{n-1}]$, un tableau d'entiers a_i de signe quelconque et de longueur arbitraire n.

1. ($\star$) Identifier une propriété qui permet de calculer de façon récursive la valeur minimale (de signe quelconque) présente dans t. Le cas terminal sera explicité. **/3**

```
Propriété récursive : min(a0, a1, ... a_n-1) = min(a0, min(a1, a2, ..., a_n-1))
Cas terminal avec la fonction min(a,b) :  min(a_i, a_j)
Cas terminal sans la fonction min(a,b) :  min(a_i, a_j) = a_i si a_i < a_j ou a_j sinon
```

2. Écrire la fonction récursive `minRec` qui calcule le minimum des valeurs d'un tableau d'entier `t` de longueur n arbitraire. Cette première version utilisera la fonction `min(a,b)` et effectuera des copies de tout ou parties de `t`. **/2**

```
fonction minRec(n : entier, t[n] tableau d'entier) retourne entier
  t2[n-1] : tableau d'entiers
debut
  si n == 2 alors
    retourne min(t[0], t[1])
  sinon
    # copie partie droite de t
    pour i de 1 a n-1 faire
      t2[i-1] = t[i]
    fin pour
    retourne min(t[0], minRec(n-1, t2))
  finsi
fin
```

3. Réécrire le traitement précédent sans utiliser la fonction `min(a,b)`. **/2**

```
fonction minRec(n : entier, t[n] tableau d'entier) retourne entier
  t2[n-1] : tableau d'entiers
debut
  si n == 2 alors
    si t[0] < t[1] alors
      retourne t[0]
    sinon
      retourne t[1]
    finsi
  sinon
    # copie partie droite de t
    pour i de 1 a n-1 faire
      t2[i-1] = t[i]
    fin pour
    val = minRec(n-1, t2)
    si t[0] < val alors
      retourne t[0]
    sinon
      retourne val
  finsi
fin

Autre version plus simple :
fonction minRec(n : entier, t[n] tableau d'entier) retourne entier
  t2[n-1] : tableau d'entiers
debut
  si n > 2 alors
    # copie partie droite de t
    pour i de 1 a n-1 faire
      t2[i-1] = t[i]
```

```
    fin pour
    retourne minRec(n-1, t2)
  sinon # la recopie garantit les indices 0 et 1 du cas terminal
    si t[0] < t[1] alors
      retourne t[0]
    sinon
      retourne t[1]
    finsi
  finsi
fin
```

4. (⋆) Réécrire le traitement précédent sans recopier le tableau `t`, ni utiliser la fonction `min(a,b)`. On pourra modifier la condition terminale et l'en-tête de cette fonction récursive. **/3**

```
Il faut que minRec sache où commence la partie droite de t
Cas terminal : deb == n-1 (la partie droite n'a plus qu'1 élément qu'il faut renvoyer).

fonction minRec2(n : entier, t[n] tableau d'entier, deb : entier) retourne entier
  val : entier
debut
  si deb == n-1 alors # il reste 1 valeur dans le sous-tab droit
    retourne t[deb]
  sinon
    val = minRec2(n, t, deb+1)
    si t[deb] < val alors
      retourne t[deb]
    sinon
      retourne val
    finsi
  finsi
fin
```

5. (⋆) Quelle est la complexité (en temps) asymptotique de cette solution ? Qu'en penser ? **/2**

```
La complexité asymptotique (en temps) est linéaire, comme celle la version itérative.
Aucune stratégie diviser pour régner n'a été appliquée ici pour que la
version récursive améliore la complexité de la version itérative.
(Non demandé: la complexité en espace de la version récursive est moins
intéressante car il faut conserver et gérer la pile des appels).
```

Exercice 3. (12 points) **/12**

On considère toujours `T` un tableau d'entiers de signe quelconque et de longueur arbitraire n. Si besoin, on peut supposer par exemple que $n \leq 100$.

1. Expliquer le principe d'une stratégie "diviser pour régner" pour calculer la valeur minimale présente dans le tableau `T`. **/3**

```
Diviser : on partage le tableau T en 2 sous-tableaux droit et gauche de longueur
moitié de celle du tableau T. Et ce récursivement jusqu'à obtenir un (sous-)tableau de
longueur 1 (un singleton).
Régner : Ce cas est terminal : on retourne directement le singleton (0
comparaison).
Combiner : On ``remonte'' cette valeur min trouvée dans le sous-tab gauche (ou
droit) qui sera comparée à l'autre trouvée dans le sous-tab droit (ou gauche),
et ce récursivement jusqu'à retrouver le tableau initial.
La valeur ainsi identifiée est la valeur minimale cherchée.
```

2. Illustrer ce principe à l'aide d'un arbre appliqué à un tableau `t8` de longueur 8. **/1**

```
a0    a1   a2   a3   a4   a5   a6   a7
   /         /         /         /
  b0         b1        b2        b3
          /                  /
       c0                  c1
                         /
                  min
```

3. (⋆) Écrire la fonction récursive `minDC` qui calcule, en appliquant la stratégie "diviser pour régner", la valeur minimale d'un tableau `t` d'entiers de longueur arbitraire `n`. **/4**

```
fonction minDC(n: entier, t[n]: tableau d'entiers, g : entier, d: entier) retourne entier
  // g et d sont les indices droits et gauche de t
  m : entier // indice du milieu
  min_g, min_d : entiers
  l : entier // longueur du sous-tab
debut
   si g == d alors
     retourner t[g]
   sinon
     m = g + (d-g)//2            // indice milieu
     l = m - g + 1               // longueur sous-tab gauche
     min_g = minDC(l, t, g, m)   // traitement récursif du sous-tab gauche
     l = d - (m+1) + 1           // longueur sous-tab droit
     min_d = minDC(l, t, m+1, d) // traitement récursif du sous-tab droit
     si min_g < min_d alors      // comparaison (combiner)
       retourne min_g
     sinon
       retourne min_d
     finsi
fin fonction
```

4. (⋆) Écrire un algorithme qui utilise `minDC` pour calculer et afficher la valeur minimale du tableau `T`. **On supposera que ce tableau est intialisé dès sa déclaration, par exemple avec la fonction `init()`.** **/3**

```
declare
  n : constante entier = 100
  T : tableau[n] d'entiers = init()
  res : entier
```

```
avec fonction minDC(n: entier, t[n]: tableau d'entiers, g : entier, d: entier)
       retourne entier
debut
  res = minDC(n, t, 0, n-1)
  afficher(``la valeur minimale de t vaut : ``, res)
fin
```

5. Combien d'appels à `minDC` sont effectués pour calculer la valeur minimale présente dans le tableau `t8` de longueur 8. **/1**

```
D'après l'arbre de la question précédente, on effectue :
1+2+4+8 = 15 appels à minDC.
```