ENSEIRB - Première Année Informatique Année 2006-2007

Initiation algorithmique

Durée : 2h00.
Notes de cours et de TD autorisées.

La complexité en temps (resp. espace) désigne la complexité en temps (resp. espace) dans le pire des cas. Lors de l'écriture de vos algorithmes, vous pourrez utiliser des fonctions auxiliaires dont vous pourrez ne pas fournir la définition algorithmique mais dont vous donnerez les spécifications (le problème résolu).

Exercice 1 L'algorithme suivant :

```
fonction calcul(T : tableau d'entiers):entier

    n ← taille(T) ;

    si n=1 ∨ n=2 alors
        retourner T[1]
    sinon
        U ← tGauche(T);
        V ← tCentre(T) ;
        W ← tDroit(T) ;

        retourner calcul(U)*calcul(V)*calcul(W) ;
```

utilise trois fonctions auxiliaires `tGauche`, `tCentre` et `tDroit` qui retournent trois tableaux d'entiers. Selon que `tGauche`, `tCentre` et `tDroit` :

1. retournent trois tableaux de tailles $\frac{n}{2}$ et ont pour complexité en temps $\theta(n^2)$.
2. retournent trois tableaux de tailles $\frac{n}{2}$ et ont pour complexité en temps $\theta(n)$.
3. retournent trois tableaux de tailles $\frac{n}{2}$ et ont pour complexité en temps $\theta(1)$.
4. retournent trois tableaux de tailles $\frac{n}{2}$, 1 et $\frac{n}{2}$ et ont pour complexité en temps $\theta(n)$.
5. retournent trois tableaux de tailles $n-1$, 1 et $n-1$ et ont pour complexité en temps $\theta(1)$.

- fournir une définition récursive de la fonction complexité en temps `calcul` notée $f(n)$
- puis une définition de $f(n)$ en fonction de n.

Exercice 2 Considérons l'algorithme suivant :

```
fonction calcul2(n:entier):entier

    n <- taille(T) ;

    si n≤ 5 alors
        retourner n
    sinon si n est pair
        retourner calcul2(n-3) + calcul2(n-4) + calcul2(n/2)
    sinon
        retourner calcul2(n-2) + calcul2(n-5) * calcul2((n-1)/2)
```

1. Quelle est la valeur de `calcul2(6)`,`calcul2(7)` ?
2. Calculer la complexité en temps de cet algorithme.
3. Ecrire un algorithme équivalent de meilleure complexité en temps.
4. Evaluer la complexité en temps de celui-ci.

Exercice 3 Considérons le problème suivant :

```
MinTableau
Entrée : deux tableaux d'entiers U et V triés sans répétition de meme taille n
Sortie : un tableau trié sans répétition de taille n contenant
         les n premiers entiers de U et de V ;
         le tableau utilise comme espace mémoire celui de U
```

Par exemple, est associé aux tableaux $U := (3, 4, 7, 8)$ et $V := (1, 2, 5, 6)$ le tableau $(1, 2, 3, 4)$.

1. Ecrire un algorithme `minTableau` de complexité en temps linéaire et de complexité en espace constant qui résolve ce problème.

Exercice 4 Considérons le problème suivant :

```
MinK
Entrée : un tableau d'entiers sans répétition  T, un entier K ≤ taille(T)
Sortie : un tableau trié de taille K contenant les K plus petites valeurs de T
```

Par exemple, est associé au tableau $T := (2, 6, 3, 8, 0, 11)$ et à l'entier 3 le tableau $(0. 2, 3)$.

1. Ecrire un algorithme récursif `minKrec` telle que l'algorithme suivant résolve `MinK` :

   ```
   fonction minK1(T : tableau d'entiers,K:entier):tableau d'entiers

       retourner minKrec(T,K,1,taille(T))
   ```

   Votre algorithme utilisera `minTableau`.
2. Fournir la définition récursive de la fonction complexité en temps de `minKrec`.
3. Evaluer la complexité en temps et en espace de `minKrec`.
4. Ecrire un algorithme `minK2` itératif de complexité en temps $O(K \cdot n)$ qui résolve `MinK`.
5. Ecrire un algorithme `minK3` de complexité en temps $O(\log(n) \cdot n)$ qui résolve `MinK`.