Université de Versailles Saint-Quentin — Algorithmique
L2 — Année 2013-2014

Contrôle continu

lundi 10 mars 2014

Toute réponse donnée doit être justifiée, en particulier sur la complexité et tout algorithme doit être expliqué au moins succinctement. L'examen dure une heure et demie et les notes prises en cours et td sont autorisées.

Exercice 1 Analyse de programme

On donne ici trois programmes dont l'entrée est un entier n. Donner la complexité dans le pire des cas de chaque programme en fonction de n. Si possible expliquer ce que renvoit le programme.

1.
```
x = 1;
Pour i = 1 à n
  j = i;
  Tant que (j<n)
    x = 2*x;
    j = j+3;
  Fin Tant que
Fin Pour
Renvoyer x
```

2.
```
x = n;
y=0;
Tant que (x >0)
  y= y+1;
  x = (x*2) % 5;
Fin Tant que
Renvoyer x + y
```

3.
```
x = n;
y = 0;
Tant que (x !=0)
  x=x/2;
  y++;
Fin Tant que
Renvoyer y;
```

Exercice 2 Tri

Un tableau d'entier T de taille n est dit bitonique si pour tout i de 0 à k les éléments sont triés en ordre croissant, puis dans un ordre décroissant de k à $n-1$:

$$T[0] \leq T[1] \leq \cdots \leq T[k] \geq T[k+1] \geq \cdots \geq T[n-1]$$

On appellera la position k le pivot du tableau bitonique.

1. Proposer un algorithme qui prend en argument un tableau bitonique et renvoie son pivot.

2. Votre algorithme résoud-il le problème en temps $O(\log(n))$.
3. Proposer un algorithme qui prend en argument un tableau bitonique T, son pivot k, et un entier x et qui détermine si x appartient au tableau en temps $O(log(n))$.
4. Proposer un algorithme qui prend en argument un tableau bitonique et le trie en temps $O(n)$.

Exercice 3 Triangle de Pascal

Le coefficient binômial $\binom{n}{p}$ vérifie la relation suivante :

$$\binom{n}{p} = \binom{n-1}{p} + \binom{n-1}{p-1}$$

Par ailleurs on sait que $\binom{n}{0} = \binom{n}{n} = 1$ pour tout n. Utiliser ces propriétés pour écrire un algorithme récursif qui étant deux entiers n et p calcule $\binom{n}{p}$.

Exercice 4 Manipulation de liste

1. Écrire un algorithme qui étant donné une liste L et un entier x :
   — recherche la première occurence de x
   — si aucune occurence n'est trouvée, ne fait rien
   — sinon déplace le premier l'élément qui contient la valeur x en tête de liste .
2. Quelle est la complexité de votre algorithme ?
3. Proposer maintenant un algorithme qui met toutes les occurences de x en tête.
4. Quelle est sa complexité ?