Université Paris 7 – Licence d'informatique IF2 – 10 avril 2009

Durée 30 minutes, tous les documents sont interdits

Nom: _____

IF 2

QCM 3, Version: A

Carte d'étudiant: _____

Remplissez la table avec les lettres correspondant à vos réponses.

Questions	1	2	3	4	5	6	7	8	9	10	11	12	13
Réponse(s)													

Bonne réponse=1pt; mauvaise réponse ou réponse incomplète =-0,5pt; pas de réponse=0pt. On rappelle qu'un arbre binaire parfait est un arbre dont tous les noeuds qui ne sont pas des feuilles ont exactement deux fils et toutes les branches (chemin de la racine à une feuille) ont la même longueur.

1. On considère le programme suivant:
   `static void f(int n){if(n==0) return; System.out.print(n); f(n-1); f(n-1)}`
   L'appel $f(4)$ affiche:
   (a) 121312141213121
   (b) 112112311211234
   (c) 432112113211211

2. On considère le programme suivant:
   `static void f(int n){if(n<=0) return; f(n-2);f(n-2);}`
   Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):
   (a) $\Theta(n)$ dans le pire cas
   (b) $\Theta(\log(n))$ dans le pire cas
   (c) $\Theta(2^n)$ dans le pire cas
   (d) infini: le programme ne termine pas toujours

3. Pour un arbre binaire parfait de hauteur n, le nombre de noeuds est:
   (a) 2^n
   (b) égal au nombre de feuilles d'un arbre parfait de hauteur $n + 1$ moins 1
   (c) $n * \log(n)$

4. Le tri fusion trie un tableau de n éléments en:
   (a) $\Theta(n^2)$ comparaisons dans le meilleur cas
   (b) $\Theta(nlog(n))$ comparaisons dans le meilleur cas
   (c) $\Theta(n)$ comparaisons dans le meilleur cas

5. Le tri par insertion trie un tableau de n éléments en:
   (a) $\Theta(n^2)$ comparaisons dans le pire cas
   (b) $\Theta(nlog(n))$ comparaisons dans le pire cas
   (c) $\Theta(n)$ comparaisons dans le pire cas

6. On considère le programme suivant:
   `static void f(int n){if(n<=0) return; if ((n%2)==0) f(n-1); else f(n+1); }`
   Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):
   (a) $\Theta(n)$ dans le pire cas
   (b) $\Theta(\log(n))$ dans le pire cas
   (c) $\Theta(2^n)$ dans le pire cas
   (d) infini: le programme ne termine pas toujours

7. Le tri rapide (quick-sort) trie un tableau de n éléments en:
   (a) $\Theta(n^2)$ comparaisons dans le pire cas
   (b) $\Theta(nlog(n))$ comparaisons dans pire cas
   (c) $\Theta(n)$ comparaisons dans le pire cas

8. On considère le programme suivant:
   `static void f(int n){if(n==0) return; f(n-1); System.out.print(n); f(n-1); }`
   L'appel $f(4)$ affiche:
   (a) 121312141213121
   (b) 112112311211234
   (c) 432112113211211

9. On considère le programme suivant:
   `static void f(int n){if(n<=1) return; f(n/2) }`
   Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):
   (a) $\Theta(n)$ dans le pire cas
   (b) $\Theta(\log(n))$ dans le pire cas
   (c) $\Theta(2^n)$ dans le pire cas

10. On considère le programme suivant:
    `static void f(int n){if(n<=1) return; f(n/2);f(n/2);}`
    Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):
    (a) $\Theta(n)$ dans le pire cas
    (b) $\Theta(n \log(n))$ dans le pire cas
    (c) $\Theta(2^n)$ dans le pire cas

11. Pour un arbre binaire parfait de hauteur n, le nombre de feuilles est:
    (a) 2^n
    (b) n
    (c) $\log(n)$

12. On considère le programme suivant:
    `static void f(int n){if(n==0) return; f(n-1); f(n-1); System.out.print(n); }`
    L'appel $f(4)$ affiche:
    (a) 121312141213121
    (b) 112112311211234
    (c) 432112113211211

13. Le tri par insertion trie un tableau de n éléments en:
    (a) $\Theta(n^2)$ comparaisons dans le meilleur cas
    (b) $\Theta(nlog(n))$ comparaisons dans le meilleur cas
    (c) $\Theta(n)$ comparaisons dans le meilleur cas

Answer Key for Exam A

Bonne réponse=1pt; mauvaise réponse ou réponse incomplète =-0,5pt; pas de réponse=0pt. On rappelle qu'un arbre binaire parfait est un arbre dont tous les noeuds qui ne sont pas des feuilles ont exactement deux fils et toutes les branches (chemin de la racine à une feuille) ont la même longueur.

1. On considère le programme suivant:
   `static void f(int n){if(n==0) return; System.out.print(n); f(n-1); f(n-1)}`
   L'appel $f(4)$ affiche:
   (a) 121312141213121
   (b) 112112311211234
   [(c)] 432112113211211

2. On considère le programme suivant:
   `static void f(int n){if(n<=0) return; f(n-2);f(n-2);}`
   Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):
   (a) $\Theta(n)$ dans le pire cas
   (b) $\Theta(\log(n))$ dans le pire cas
   [(c)] $\Theta(2^n)$ dans le pire cas
   (d) infini: le programme ne termine pas toujours

3. Pour un arbre binaire parfait de hauteur n, le nombre de noeuds est:
   (a) 2^n
   [(b)] égal au nombre de feuilles d'un arbre parfait de hauteur $n + 1$ moins 1
   (c) $n * \log(n)$

4. Le tri fusion trie un tableau de n éléments en:
   (a) $\Theta(n^2)$ comparaisons dans le meilleur cas
   [(b)] $\Theta(nlog(n))$ comparaisons dans le meilleur cas
   (c) $\Theta(n)$ comparaisons dans le meilleur cas

5. Le tri par insertion trie un tableau de n éléments en:
   [(a)] $\Theta(n^2)$ comparaisons dans le pire cas
   (b) $\Theta(nlog(n))$ comparaisons dans le pire cas
   (c) $\Theta(n)$ comparaisons dans le pire cas

6. On considère le programme suivant:
   `static void f(int n){if(n<=0) return; if ((n%2)==0) f(n-1); else f(n+1); }`
   Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):
   (a) $\Theta(n)$ dans le pire cas
   (b) $\Theta(\log(n))$ dans le pire cas
   (c) $\Theta(2^n)$ dans le pire cas
   [(d)] infini: le programme ne termine pas toujours

7. Le tri rapide (quick-sort) trie un tableau de n éléments en:
   [(a)] $\Theta(n^2)$ comparaisons dans le pire cas
   (b) $\Theta(nlog(n))$ comparaisons dans pire cas
   (c) $\Theta(n)$ comparaisons dans le pire cas

8. On considère le programme suivant:
   `static void f(int n){if(n==0) return; f(n-1); System.out.print(n); f(n-1); }`
   L'appel $f(4)$ affiche:
   [(a)] 121312141213121
   (b) 112112311211234
   (c) 432112113211211

9. On considère le programme suivant:
   `static void f(int n){if(n<=1) return; f(n/2) }`
   Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):

   (a) $\Theta(n)$ dans le pire cas
   [(b)] $\Theta(\log(n))$ dans le pire cas
   (c) $\Theta(2^n)$ dans le pire cas

10. On considère le programme suivant:
    `static void f(int n){if(n<=1) return; f(n/2);f(n/2);}`
    Le nombre d'appels de f pour $f(n)$ est un (on suppose $n > 0$):

    [(a)] $\Theta(n)$ dans le pire cas
    (b) $\Theta(n \log(n))$ dans le pire cas
    (c) $\Theta(2^n)$ dans le pire cas

11. Pour un arbre binaire parfait de hauteur n, le nombre de feuilles est:

    [(a)] 2^n
    (b) n
    (c) $\log(n)$

12. On considère le programme suivant:
    `static void f(int n){if(n==0) return; f(n-1); f(n-1); System.out.print(n); }`
    L'appel $f(4)$ affiche:

    (a) 121312141213121
    [(b)] 112112311211234
    (c) 432112113211211

13. Le tri par insertion trie un tableau de n éléments en:

    (a) $\Theta(n^2)$ comparaisons dans le meilleur cas
    (b) $\Theta(nlog(n))$ comparaisons dans le meilleur cas
    [(c)] $\Theta(n)$ comparaisons dans le meilleur cas