13,5/20

NOM : CAPOSTOLET Prénom : Anaïs Groupe : 112

IUT Paris Descartes – Dép. Informatique – Analyse et Méthodes numériques S2 – DST du 11 Avril 2018

Sujet A

Sans calculatrice, sans portable, sans document. – Durée : 1h.

ATTENTION : On répondra sur le sujet, en complétant dès le début de l'épreuve vos noms, prénoms et groupe. Une feuille de brouillon est disponible à la fin du sujet.

Exercice 1 (10 points)

Dans cet exercice, vous répondrez directement sur le sujet. Pour chaque question, une ou plusieurs réponse(s) sont possibles. Une réponse juste rapporte les points correspondants, une réponse fausse enlève la moitié des points.

1. $n^3 \ln n + n^2 = \Theta(n^4)$

   ☐ vrai ☒ faux

   Justifier : $\lim_{n\to+\infty} n^3 \ln n + n^2 = \lim_{n\to+\infty} n^3 \ln n$

   $\lim_{n\to+\infty} \frac{n^3 \ln n}{n^4} = 0$ On est une limite finie donc $n^3 \ln n + n^2 = O(n^4)$

   $\lim_{n\to+\infty} \frac{n^4}{n^3 \ln n} = \lim_{n\to+\infty} \frac{n^4}{n^3} = \lim_{n\to+\infty} n = +\infty$

   $+\infty$ est une limite infinie donc $n^3 \ln n + n^2 \neq O(n^3) \neq \Theta(n^4)$

2. $\lim_{n\to+\infty} n^2(\ln n)^3 - n^3(\ln n)^2$

   ☐ $= +\infty$ ☒ $= -\infty$ ☐ $= 0$ ☐ n'existe pas

   Justifier : $\lim_{n\to+\infty} n^2(\ln n)^3 - n^3(\ln n)^2 = \lim_{n\to+\infty} n^3[(\ln n)^3 - n(\ln n)^2]$

   On $\lim_{n\to+\infty} [(\ln n)^3 - n(\ln n)^2] = -\infty$ (car $n \gg \ln n$)

   Donc $\lim_{n\to+\infty} n^2[(\ln n)^3 - n(\ln n)^2] = -\infty$

3. Cochez les affirmations vraies lorsque n tend vers l'infini :
☒ $4n^3 + n\ln n + 2 = O(n^3)$ ☒ $4n^3 + n\ln n + 2 = \Theta(n^3)$
☐ $4n^3 + n\ln n + 2 = o(n^3)$ ☐ $4n^3 + n\ln n + 2 \sim n^3$

4. Cochez les affirmations vraies lorsque n tend vers l'infini :
☒ $2^n \ln n + n^2 = O(e^n)$ ☐ $2^n \ln n + n^2 = \Theta(3^n)$
☐ $2^n \ln n + n^2 \sim 2^n$ ☒ $2^n \ln n + n^2 = o(2^n)$

5. La complexité du problème du voyageur de commerce (ou recherche d'un chemin Hamiltonien) dans un graphe à n sommets se mesure actuellement en :
☐ $\Theta(2^n)$ ☐ $\Theta(e^n)$ ☒ $\Theta(n!)$ ☐ $\Theta(n^2)$

6. A l'heure actuelle, la meilleure complexité des algorithmes effectuant le produit de deux matrices de taille n est de type :
☐ linéaire ☒ logarithmique ☐ polynomiale ☐ exponentielle

7. *(Bonus)* Logarithme et Exponentielle, vont au restaurant. Qui paie l'addition ?
☐ Logarithme ☒ Exponentielle
Justifier : Car une complexité exponentielle coute plus cher à exécuter en machine ?

Exercice 2 **(3 points)** On considère le script suivant dans Scilab.

```
1  A=rand(n,n);
2  S=0;
3  for i=1:n
4      for h=1:n
5          S=S+A(i,h);
6      end
7  end
8  disp S
9      end
10 end
```

1. Combien de fois l'algorithme exécute-t-il la commande de la ligne 5 ? On a deux boucles for imbriquées avant de la ligne 5 qui s'exécute $(n-1)^2$ fois
n^2

2. En déduire le nombre total $T(n)$ d'instructions exécutées par l'algorithme. On a deux instructions au début et une à la fin et les affectations de boucles, on a donc :
$2(n-1)^2 + 3 = 2(n^2 - 2n + 1) + 3$
$= 2n^2 - 4n + 5$

3. Cet algorithme est-il de complexité linéaire ? Justifier. L'expression $2n^2 - 4n + 5$ étant un polynôme on peut dire que cet algorithme a une complexité polynomiale et non linéaire.

/ 8

Exercice 3 (7 points)

On propose les quatre scripts suivants, dont le but est, pour une fonction f donnée, de renvoyer une solution approchée de l'équation $f(x) = 0$ dans l'intervalle $[a, b]$ avec une erreur e :

```
function [u,n]=Script1(a,b,f,e)
    u=a;
    v=b;
    n=0;
    while (abs(u-v)>e)
        v=u;
        u=u-f(u)*(b-u)/(f(b)-f(u));
        n=n+1;
    end
endfunction
```

```
function [u,n]=Script2(a,b,f,e)
    u=0;
    v=b;
    n=0;
    while (abs(u-v)>e)
        v=u;
        u=u-f(u)*(a-u)/(f(a)-f(u));
        n=n+1;
    end
endfunction
```

```
function [u,n]=Script3(a,b,f,e)
    u=a;
    v=b;
    n=0;
    while (abs(u-v)<e)

        if f(u)*f((u+v)/2)>0 then
            u=(u+v)/2;
        else
            v=(u+v)/2;
        end
        n=n+1;
    end
endfunction
```

```
function [u,n]=Script4(a,b,f,e)
    u=a;
    v=b;
    n=0;
    while abs(u-v)>e
        m=(u+v)/2;
        if f(u)*f(m)<0 then
            .........
        else
            .........
        end
        n=n+1;
    end
endfunction
```

1. Quelles hypothèses doivent vérifier la fonction f si l'on désire résoudre l'équation $f(x) = 0$ par la méthode de Newton ? La fonction f doit avoir une dérivée qui ne s'annule jamais sur l'intervalle étudié pour résoudre l'équation f(x)=0 par la méthode de Newton.

1

2. Pour appliquer la méthode de la sécante à la fonction f dont la courbe est donnée en annexe, quel script doit-on exécuter ? pour la méthode de la sécante, il faut utiliser le script

3. Construire sur le graphique en annexe les quatre premiers termes (u_0, u_1, u_2, et u_3) de la suite correspondant à la méthode de la sécante, en prenant pour $a = 4$ et $b = 9$ ~~les valeurs de la question 1.b.~~ (On mentionnera clairement les termes u_0, u_1, u_2, et u_3 sur le graphique, ainsi que les traits de construction). de l'annexe

4. En appliquant le script 3 à la fonction ~~de la question 1.b~~, on obtient en sortie $n = 0$. Que cela signifie-t-il ? Corriger le script en conséquence pour que celui-ci correspondent à la méthode par dichotomie.

cela signifie que l'algorithme ne rentre pas dans la boucle.

Il faut corriger la condition de la boucle en :

while abs(u-v)>e

5. Comment doit-on compléter les lignes 46 et 48 du script 4 pour que celui-ci correspondent à la méthode par dichotomie ?

Ligne 46 : v = m ;

Ligne 48 : u = m ;