TD Algorithmique - Feuille 2

Complexité des algorithmes itératifs

Notations asymptotiques

1. Caractériser les fonctions suivantes en terme de notation θ :
   - $f(n) = 4n^3 + n$
   - $f(n) = \frac{n(n+1)}{2}$
   - $f(n) = n * log_2(n)$
2. Parmi les fonctions suivantes, quelles sont celles qui ont le même ordre de grandeur ?
   - $f_1(n) = 4n^3 + n$
   - $f_2(n) = n^2 + log_2(n)$
   - $f_3(n) = n^2 * log_3(n) + 6n^3$
   - $f_4(n) = \frac{n(n^2+1)}{2}$
3. En utilisant la définition de θ montrer que $f(n) + g(n) = \theta(max(f(n), g(n)))$
4. En utilisant la définition de θ montrer que $3n^2 + 4n + 6 = \theta(n^2)$

Complexité

5. Exprimer en fonction de n le nombre de fois où s'affiche "Bonjour" dans les deux morceaux de code suivants :

```
// Code 1
for (int i=0;i<n;i++)
    for (int j=0;j<n;j++)
        System.out.println(‘‘Bonjour’’);
// Code 2
for (int i=0;i<n;i++)
    for (int j=0;j<i;j++)
        System.out.println(‘‘Bonjour’’);
```

   Ces deux codes ont ils la même complexité ?

6. Calculer la complexité de la méthode "public boolean palindrome(String c)".
7. Caractériser la complexité de la méthode "public static int recherche(int[] tab, int x, int o)" de la classe "TableauEntier" en utilisant les notations Ω, θ et O.

8. On s'intéresse à la méthode de tri appelée "tri à bulle". Compléter les assertions A1, A2 et A3 et expliquer comment fonctionne ce tri. Calculez la complexité dans le pire et le meilleur des cas de la méthode "tri bulle". Donner des exemples de tableau pour lesquels ces complexités sont atteintes.

```
/** ANTECEDENT : tab est un tableau d'entiers
 * CONSEQUENT : tab est trié par ordre croissant */
public static  void triBulle(int[] tab) {
    boolean fini = false;
    int j = tab.length -1;
    while (!fini) {
        fini=true;
        for (int i=0; i<j; i++) {
            if (tab[i]>tab[i+1]) {
                int tmp = tab[i];
                tab[i] = tab[i+1];
                tab[i+1] = tmp;
                fini = false;
            }
        }
        // A1 : caractériser tab[j]
        // A2 : quelle est la partie du tableau qui est triée ?
        // A3 : comparer les éléments de 0 à j-1 par rapport aux éléments de j à tab.length-1
        j--;
    }
 }
```

9. Calculer la complexité de la méthode "public static int mystere(int x)".

10. Calculer la complexité dans le pire des cas de la méthode "rechercheVite" du cours (compter le nombre de passages dans la boucle while).

11. Pour crypter des messages on peut utiliser un algorithme qui consiste à ajouter à chaque caractère du message les caractères d'une clef. Par exemple, si le message est "bonjour", la clef est "gag" on obtient "ÉΔÕÑΔÜÙ".

    b o n j o u r
    + a g a g g a g
    = ÉΔÕÑΔÜÙ

    Comparer les méthodes "code" et "decode" de la classe "Crypto" en annexe.

    Ces méthodes ont-elles la même complexité ?

Annexe

```
CLASSE CRYPTO
-------------
/** cryptage selon l'algorithme de Vigénère 16ème sciècle*/

public class Crypto {
    private String cle; // la clef du cryptage
    // constructeur
    public Crypto(String s) {
        cle= s;
    }
    // encodage
    public String code(String s) {
        char[] result = new char[s.length()];// tableau contenant les caractères
        int i = 0;
        while (i<s.length())
            for (int j=0; j < cle.length()&& i < s.length(); j++) {
                // on fait la somme modulo 255 car le code ASCII est sur 8 bits (de 0 à 255)
                result[i] = (char)((s.charAt(i) + cle.charAt(j))%255) ;
                i++;
            }
        return new String(result);
    }
    // décodage
    public String decode(String s) {
        char[] result = new char[s.length()]; // tableau contenant les caractères
        int i = 0;
        int j = 0;
        int l = cle.length();
        while (i<s.length()){
            // on fait la somme modulo 255 car le code ASCII est sur 8 bits (de 0 à 255)
            result[i] = (char)((s.charAt(i) - cle.charAt(j))%255) ;
            i++;
            j = (j+1) % l;
            }
        return new String(result);
    }
}
```