Université Batna2 — Master 1 CS
Département Informatique — TD1
Module : Complexité algorithmique et chiffrement

Exercice 1 :

Etudier la complexité de l'exemple ci-dessous :

Fonction de multiplication de deux matrices[4]

```
MULTIPLICATIONMATRICES(A,B)
entrée : deux matrices A, B n×n
sortie : matrice C n×n
n ← ligne[A]
Soit C une matrice n×n
pour i ← 1 à n
    faire pour j ← 1 à n
        faire c_ij ← 0
            pour k ← 1 à n
                faire c_ij ← c_ij + a_ik . b_kj
            fin pour
    fin pour
fin pour
retourner C
```

Exercice 2 :

Donnez la complexité de l'algorithme suivant :

Fonction factoriel calculée par récursivité

```
FACTORIEL (n)
entrée : un entier n
sortie : n!
Si n ≤ 1
    alors retour ← 1
    sinon retour ← n × FACTORIEL(n−1)
fin si
retourne retour
```

Exercice 3 :

Calculer la complexité en meilleur et pire de cas de l'algorithme tri à bulle suivant :

```
TRI-BULLE (A)
entrée : tableau A
sortie : tableau A trié par ordre croissant
Pour i ← 1 à longueur[A]
    faire pour j ← longueur[A] décroissant jusqu'à i+1
        faire si A[j] < A[j-1]
            alors permuter A[j] ↔ A[j – 1]
        fin si
    fin pour
fin pour
```

Exercice 4 :

Calculer la complexité de l'algorithme de recherche dichotomique suivant :

```
RECHERCHE_DICHOTOMIQUE
Entrée : un tableau de n nombres A=[a1,a2,a3,...,an] trié par ordre croissant
et une valeur v
Sortie : un indice i tel que v=A[i], ou bien la valeur spéciale NIL si v ne
figure pas dans A
début ← 1
fin ← n
trouvé ← FALSE
répéter
    milieu ← partie entière(début + (fin - début) / 2)
    si A[milieu]=v
    alors trouvé ← TRUE
    sinon
        si v≥A[milieu]
        alors
            début ← milieu + 1
            si A[début]=v
            alors
                milieu ← début
                trouvé ← TRUE
            fin si

        sinon
            fin ← milieu - 1
            si A[ fin]=v
            alors
                milieu ← fin
                trouvé ← TRUE
            fin si
        fin si
    fin si
jusqu'à trouvé ou début≥ fin
```

Exercice 5 :

Quelle est la complexité des fonctions suivantes :

- $f(n)=5n^4+3n^3+2n^2+4n+1$
- $f(n)=5n^2+3n\log n+2n+5$
- $f(n)=3\log n+2/n$
- $f(n)=2^{n+2}$

Solutions des Exercices

Exo1 :

- *Multiplication de deux matrices :* L'algorithme est *l'exemple 1*

Soient $T(n)$ le temps d'exécution en fonction de l'argument n et c_i le coût en temps de la ligne i.

Nous avons : $T(n)=c_1+c_2+n[c_3+n(c_4+c_5+n(c_6+c_7))]$

$$T(n)=c_1+c_2+c_3.n+(c_4+c_5).n^2+(c_6+c_7).n^3$$

Ainsi $T\in\Theta(n^3)$ pour les trois complexités.

Exo2 :

Factoriel :

Soit $T(n)$ *le temps d'exécution en fonction de l'argument* n.
À la base $T(1)=c_1+c_2$
Sinon $T(n)=c_1+c_3+T(n-1)$ *pour* $n>1$
Démontrons que $T(n)=c_1+c_2+(n-1)(c_1+c_3)$ *pour* $n\geqslant 1$

Pour $n=1$ *: vrai.*
Nous le supposons vrai au degré n.
Démontrons-le au degré $n+1$.

$$T(n+1)=c_1+c_3+T(n)$$

$$T(n+1)=c_1+c_3+c_1+c_2+(n-1)(c_1+c_3)=c_1+c_2+n(c_1+c_3)$$

Comme la proposition est vraie au degré 1, et si elle est vraie au degré n, *elle est vraie au degré* $n+1$, *d'après le principe de récurrence, elle est vraie quel que soit* n.

Donc $T(n)=\Theta(n)$

Exo3 : (tri à bulle)

Calcul de la complexité dans le meilleur des cas : le tableau est déjà trié par ordre croissant. La quatrième ligne n'est jamais exécutée.

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n}(c_1 + \sum_{j=i+1}^{n}(c_2 + c_3)) \\
&= \sum_{i=1}^{n} c_1 + \sum_{i=1}^{n}\sum_{j=i+1}^{n}(c_2 + c_3) \\
&= c_1 . \sum_{i=1}^{n} 1 + (c_2 + c_3) . \sum_{i=1}^{n}\sum_{j=i+1}^{n} 1 \\
&= c_1 . n + (c_2 + c_3) . \sum_{i=1}^{n}(n-i)
\end{aligned}
$$

$$
\begin{aligned}
\sum_{i=1}^{n}(n-i) &= \sum_{i=1}^{n} n - \sum_{i=1}^{n} i \\
&= n^2 - \frac{1}{2} n(n+1)\,^{*} \\
&= n^2 - \frac{1}{2} n^2 - \frac{1}{2} n \\
&= \frac{1}{2} n^2 - \frac{1}{2} n
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= c_1 . n + (c_2 + c_3) . (\frac{1}{2} n^2 - \frac{1}{2}) \\
&= \frac{(c_2 + c_3)}{2} n^2 + \frac{2c_1 - c_2 - c_3}{2} n
\end{aligned}
$$

D'où $T(n) \in \Theta(n^2)$

* Rappel : la sommation d'une série arithmétique a pour valeur $\sum_{k=1}^{n} k = \frac{1}{2} n(n+1)$

Calcul de la complexité dans le pire des cas : le tableau est trié par ordre décroissant

La quatrième ligne est toujours exécutée. Nous avons donc :

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n}(c_1 + \sum_{j=i+1}^{n}(c_2 + c_3 + c_4)) \\
&= c_1 . n + (c_2 + c_3 + c_4) . (\frac{1}{2} n^2 - \frac{1}{2}) \\
&= \frac{(c_2 + c_3 + c_4)}{2} n^2 + \frac{2c_1 - c_2 - c_3 - c_4}{2} n
\end{aligned}
$$

D'où $T(n) \in \Theta(n^2)$

L'algorithme tri à bulle est donc dans tous les cas de complexité $\Theta(n^2)$.

Exo4 :

Calcul de la complexité dans le meilleur des cas : l'élément $A[milieu]=v$.

$T(n)$ vaut donc :
$T(n)=c_1+c_2+c_3+c_4+c_5+c_6+c_7+c_{25}+c_{26}$
$T(n)=\Theta(1)$

Calcul de la complexité dans le pire des cas : v ne figure pas dans A .

Supposons que v est plus grand que tous les éléments de A .
$T(n)$ vaut donc :

$$T(n)=c_1+c_2+c_3+x.(c_4+c_5+c_6+c_8+c_9+c_{10}+c_{11}+c_{12}+c_{16}+c_{24}+c_{25}+c_{26})$$

L'algorithme passe x fois dans la boucle répéter jusqu'à ce que $début \geq fin$

Calculons x. Pour cela, nous allons émettre une hypothèse puis la démontrer par récurrence.
$T(n)$ est du type : $T(n)=C_1+C_2x$ avec C_1, C_2 des constantes.
Si $n=2$ ou $n=3$, $x=1$.
Si $n=4$ ou $n=5$, $x=2$.
Si $n=8$ ou $n=9$, $x=3$.
En fait, C_1 est le temps pour initialiser l'algorithme et C_2x le temps nécessaire à l'algorithme pour gérer le tableau de taille n . Posons $T'(n)$ le temps pour effectuer la boucle pour une taille n .
Par la suite, pour simplifier, nous supposons que la taille du problème initial est une puissance de deux.
Démontrons que $T'(n)=C_2.\log_2(n)$.
Nous le supposons vrai … au niveau n .

Démontrons-le au niveau $2n$.
$T'(2n)=C_2+T'(n)=C_2+C_2.\log_2(n)=C_2(1+\log_2(n))=C_2(\log_2(2)+\log_2(n))=C_2\log_2(2n)$
Comme la proposition est vraie au degré 1, et si elle est vraie au degré n , elle est vraie au degré $2n$, d'après le principe de récurrence, elle est vraie quel que soit n puissance de 2.
D'où

$$T(n)=C_1+T'(n)=C_1+C_2.\log_2(n)\in\Theta(1)+\Theta(\log_2(n))$$

D'où $T(n)\in\Theta(\log_2(n))$

Exo5 :

Calcule de la complexité des fonctions :

1) f(n)∈O(n^4)
2) f(n)∈O(n^2)
3) f(n)∈O($\log(n)$)
4) f(n)∈O(2^n)