ALGORITHMIQUE II

SMI - S3

NOTION DE COMPLEXITE

Partie II

Département d'Informatique
FSDM-USMBA-FES

Etapes à suivre.

- Choisir le paramètre qui correspond à la taille du problème: n

- Identiier les instructions basiques de l'algorithme:
  celles qui contribuent considérablement dans le temps d'exécution.

- Déterminer les cas: pires et meilleurs pour des données de taille n

- Evaluer une somme pour le nombre de fois que les opérations de base sont exécutées.

- Déduire l'ordre de grandeur de cette somme.

Formules de sommation et règles utiles.

- $$\sum_{i=1}^{n} 1 = 1 + 1 + \cdots + 1 = n \in \Theta(n)$$
- $$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \in \Theta(n^2)$$
- $$\sum_{i=1}^{n} i^k = 1 + 2^k + \cdots + n^k \approx \frac{n^{k+1}}{k+1} \in \Theta(n^{k+1})$$
- $$\sum_{i=1}^{n} a^i = 1 + a + \cdots + a^n = \frac{a^{n+1}-1}{a-1} \in \Theta(a^n)$$
- $$\sum_{i=1}^{n} (a_i \pm b_i) = \sum_{i=1}^{n} a_i \pm \sum_{i=1}^{n} b_i \qquad \sum_{i=1}^{n} c\, a_i = c \sum_{i=1}^{n} a_i$$

Exemple1. Recherche du Maximum: O(n)

```
algorithme MaxElement(A[0..n- 1])
// Retourne la valeur maximale dans un tableau
// Entrée: Un tableau non-vide de réels
// Sortie: le maximum dans A
maxval ⟵ A[0]
Pour i allant de 1 jusqu'à n - 1 faire
        si A[i] > maxval alors
                maxval ⟵ A[i]
finPour
retourner maxval
```

Opération de base: la comparaison dans la boucle “Pour”

L'algorithme effectue $\sum_{i=1}^{n-1} 1 = n - 1$ comparaisons

Exemple2. Test d'éléments uniques: $O(n^2)$

```
algorithme UniqueElements (A[0..n- 1])
// Retourne Vrai si tous les éléments sont différents
// Entrée: Un tableau A
// Sortie:retourner VRAI s'il n y a pas de doublons
Pour i allant de 0 jusqu'à n - 2 faire
        Pour j allant de i + 1 jusqu'à n - 1 faire
                si A[i] = A[j] alors retourner FAUX
        finPour
 finPour
retourner VRAI
```

Opération de base: la comparison dans la boucle imbriquée

L'algorithme effectue dans le meilleur des cas: 1 comparaison

et dans le pire des cas $\sum_{i=0}^{n-2} n-(i+1)=(n-1)+(n-2)+\cdots+1=\sum_{k=1}^{n-1} k=n(n-1)/2$

Exemple3. Produit matriciel: $O(n^3)$

```
Algorithme      MatProduit (A,B : n × n matrices)
// Multiplie la matrice A fois la matrice B
// Entrée: Deux matrices (n × n) A et B
// Sortie: Matrice C = AB
Pour i ← 0 jusqu'à n – 1 faire
          Pour j ← 0 jusqu'à n – 1 faire
                    C[i, j] ← 0
                  Pour k ← 0 jusqu'à n – 1 faire
                              C[i, j] ← C[i, j] + A[i, k] ∗ B[k, j]
                    finPour
          finPour
FinPour
retourner C
```

Opération de base: la multiplication dans la boucle imbriquée

L'algorithme effectue $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} 1 = n^3$ multiplications.

Exemple4. Nombre de bits dans un entier: O(n)

```
Algorithme BitCount(m)
// Entrée: Un entier positif m
// Sortie: Le nombre de bits pour coder m
   count ← 1
Tant que m > 1 faire
   count ← count + 1
   m ← ⌊m/2⌋ // division entière
retourner count
```

n= nombre de bits dans l'entier m

Opération de base:la division par 2 dans la boucle

L'algorithme effectue $n - 1$ opérations puisque $n = count$.

Etapes à suivre.

- Choisir le paramètre qui correspond à la taille du problème: n

- Identiier les instructions basiques de l'algorithme:
  celles qui contribuent considérablement dans le temps d'exécution.

- Déterminer les cas: pires et meilleurs pour des données de taille n

- Etablir une relation de récurence exprimant le compte des opérations de base entre deux appels récursifs consécutifs.

- Résoudre la récurence en fonction de n (Déduire l'ordre de grandeur).

Exemple5. factoriel récursif: O(n)

```
Algorithme Factoriel(n)
// calcul récursif de n!
// Entrée: un entier positif ou nul: n
// Sortie: la valeur de n!
   si n = 0 alors retourner 1
   sinon retourner Factoriel(n − 1) ∗ n
```

Taille du problème: l'entier n

Opération de base: les miltiplications

Exemple5. factoriel récursif: O(n)

- soit $M(n)$ = le nombre de multiplications à effectuer pour calculer *Factoriel(n)*.
- $M(0) = 0$ car aucune multiplication n'est effectuée pour évaluer *Factoriel(0)*.
- si $n > 0$, alors *Factoriel(n)* effectue un appel recursif plus une multiplication.

$$M(n) = \underbrace{M(n-1)}_{\text{pour calculer Factoriel(n-1)}} + \underbrace{1}_{\text{pour multiplier Factoriel(n-1) par n}}$$

Exemple5. factoriel récursif: O(n)

$$M(0) = 0$$
$$M(n) = M(n-1) + 1$$

On peut remarquer que M(n) est une suite arithmétique de raison r=1

$$M(n) = n\,r + M(0)$$

Dans ce cas simple on peut facilement montrer (par induction) que:

$$M(n) = n$$

Preuve par induction:

- base: si $n = 0$, alors $M(n) = M(0) = 0 = n$
- Induction: si $M(n-1) = n-1$, alors
  $M(n) = M(n-1) + 1 = (n-1) + 1 = n$

Exemple6. tours de Hanoi: $O(2^n)$

```
Algorithme Towers(n, i, j)
// Déplace n disques du piquet i au piquet j
// Entrée: les entiers n > 0, 1 ≤ i, j ≤ 3, i ≠ j
// Sortie: Spécifier les déplacements des
//           disques dans un ordre correct.

    si n = 1 alors
        Ecrire (« déplacer 1 disque de » , i, « → », j)
    sinon
        Towers(n−1, i, 6−i−j)
        Ecrire (« déplacer 1 disque de » , i, « → », j)
        Towers(n−1, 6−i−j, j)
    finsi
```

Taille du problème: l'entier n représentant le nombre de disques à déplacer

Opération de base: déplacer des disques

Exemple6. tours de Hanoi: $O(2^n)$

- soit $M(n)$ = le nombre de déplacements à effectuer pour exécuter Towers(n,.,.).
- $M(1) = 1$ car aucune *1 déplacement est nécessaire pour exécuter Towers(1,.,.)*
- si $n > 1$, alors *Towers(n,.,.) effectue 2 appels récursifs plus un déplacement*.

$$M(n) = \underbrace{2M(n-1)}_{\text{pour exécuter } Towers(n-1,\ \cdot,\ \cdot) \text{ 2 fois}} + \underbrace{1}_{\text{pour déplacer 1 disque}}$$

Exemple6. tours de Hanoi: $O(2^n)$

- *substitution en ordres croissants* $M(2) = 2M(1) + 1 = 3$

$$M(3) = 2M(2) + 1 = 7$$
$$M(4) = 2M(3) + 1 = 15$$

- *substitution en ordres décroissants* $M(n) = 2M(n-1) + 1$

$$= 2[2M(n-2) + 1] + 1 = 4M(n-2) + 3$$
$$= 4[2M(n-3) + 1] + 2 = 8M(n-3) + 7$$

Preuver $M(n) = 2^n - 1$ *par induction*

- *base : si* $n = 1$, *alors* $M(n) = 1 = 2^n - 1$
- Induction: *si* $M(n-1) = 2^{n-1} - 1$, *alors*

$$M(n) = 2M(n-1) + 1 = 2 * (2^{n-1} - 1) + 1 = (2^n - 2) + 1 = 2^n - 1$$