TD 5 - La complexité

Inès de Courchelle

2023-2024

Objectifs :

- Améliorer les algorithmes
- Mesurer le coût des algorithmes
- Comprendre les processus d'élaboration algorithmique

Durée 3h
Format papier

Introduction

Objectifs d'un Algorithme

- Répondre à un problème/besoin
- Être rapide (complexité temporelle)
- Utiliser peu de mémoire (complexité spatialle)

Objectifs de la complexité

- Comparer et mesurer les performances d'algorithmes répondant à un même problème
- Réduire le temps d'exécution d'algorithme
- Utiliser un minimum de la mémoire

Les différents types de complexité

Rappel du cours

La complexité en nombre de comparaisons

- C'est le nombre de comparaisons réalisées dans un algorithme.
- En C, une comparaison est donnée par les opérateurs de comparaison suivant : ==, <, >, !=, <=, >=

Cas particuliers :

- Une instruction de boucle est égale à N+1-i comparaison pour chaque variable d'itération i
- Une condition est égale à 1 comparaison

Exemple 1

```
N <- 10
POUR i allant de 0 à N FAIRE
    ...
FIN POUR
```

$N + 1 - i = 10 + 1 - 0 = 11$ comparaisons

Exemple 2

```
N <- 10
i<- 0
TANT QUE (i<N) FAIRE
    i <- i++
FIN TANT QUE
```

$N + 1 - i = 10 + 1 - 0 = 11$ comparaisons

Exemple 3

```
SI (a<10) ALORS
    ...
FIN SI
```

1 comparaison

Exemple 4

```
SI (a=2 ET b=4) ALORS
```

```
    ...
FIN SI
```

2 comparaisons

La complexité en nombre d'opérations

- C'est le nombre de calcul réalisé dans un algorithme.
- En C, une opération est donnée par les opérateurs suivant : +, -, ++, –, *

Cas particulier : Une instruction de boucle **for** réalise N-i opérations pour chaque variable d'itération i.

Exemple 1

```
N <- 10
POUR i allant de 0 à N FAIRE
    ...
FIN POUR
```

$N - i = 10 - 0 = 10$ opérations

Exemple 2

```
N <- 10
POUR i allant de 4 à N FAIRE
    ...
FIN POUR
```

$N - i = 10 - 4 = 6$ opérations

La complexité en nombres d'affectations

- C'est le nombre d'écriture réalisée dans la mémoire.
- En C, une affectation est donnée par l'unique opérateur suivant : =

Cas particuliers : Une instruction de boucle **for** réalise N+1-i opérations pour chaque variable d'itération i.

Exemple 1

```
N <- 10
```

```
POUR i allant de 0 à 10 FAIRE
    ...
FIN POUR
```

$N + 1 - i = 10 + 1 - 0 = 11$ affectations

Exemple 2

```
N <-10
POUR i allant de 4 à 10 FAIRE
    ...
FIN POUR
```

$N + 1 - i = 10 + 1 - 4 = 7$ affectations

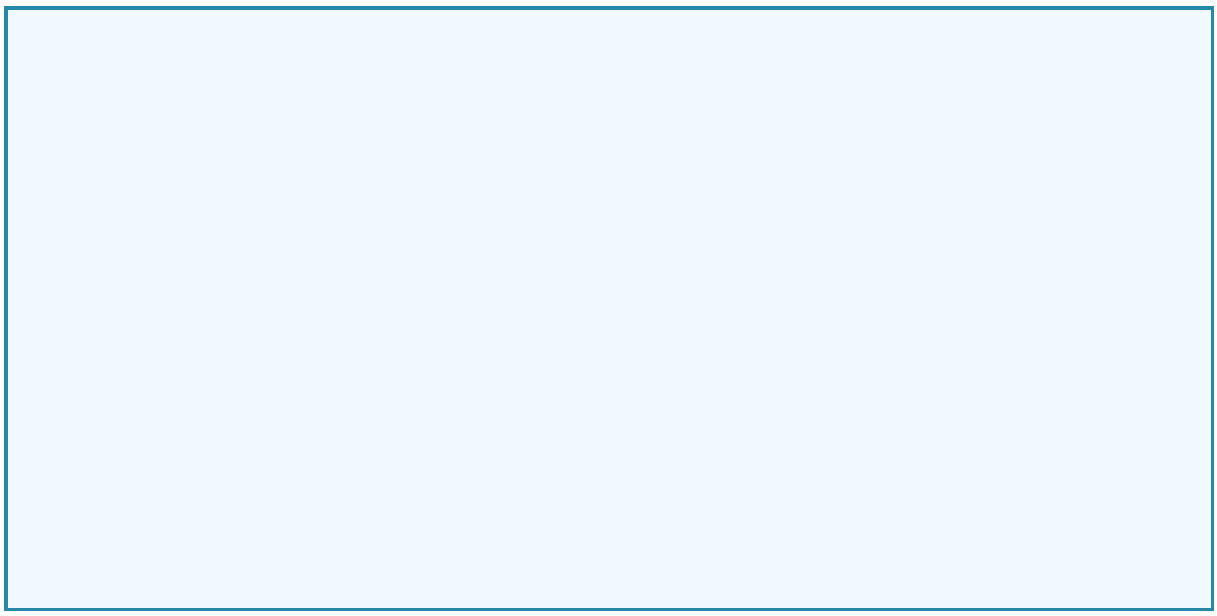
Exercices

Exercice 1

Nous considérons l'algorithme donné par le code suivant :

```
i <- 5
TANT QUE (i > 0) FAIRE
    s <- 0
    POUR j allant de 0 à i FAIRE
        s <- s+1
    FIN POUR
    i <- i -1
FIN TANT QUE
```

Calculer sa complexité en nombre de comparaison, d'affectation et d'opération.



Exercice 2

Les deux algorithmes suivants permettent de calucler les calculs pour Calculer X^{2N}.

Algorithme 1

```
X2N <- 1
POUR i allant de 0 à (2*N) FAIRE
    X2N <- X2N * X
FIN POUR
```

Algorithme 2

```
X2N <- 1
X2 <- X*X
POUR i allant de 0 à N FAIRE
  X2N <- X2N*X2
FIN POUR
```

Calculer sa complexité en nombre de comparaison,d'affectation et d'opération.

Les classes de complexité

Rappel du cours

- La complexité ne prend en compte qu'un ordre de grandeur d'opération.
- Pour représenter cette approximation, la notation est => $\mathcal{O}$
- Pour dire qu'une méthode s'effectue environ en N^2 opérations, on dit qu'elle a une complexite $\mathcal{O}(N^2)$

Exemple 1 : $2 \times N + 5$ opérations

$\mathcal{O}(N)$

Exemple 2 : $\frac{N}{2}$ opérations

$\mathcal{O}(N)$

Exemple 3 : $2 \times N^2 + 3N + 5$ opérations

$\mathcal{O}(N^2)$

Exercices

Exercice 1

Donner l'ordre de complexité des algorithmmes suivants (en fonction des variables n et m) :

```
x <- 1
POUR i allant de 1 à n FAIRE
  POUR j allant de 1 à m FAIRE
      x <- x+1
  FIN POUR
FIN POUR
```

Exercice 2

Donner l'ordre de complexité des algorithmmes suivants (en fonction des variables n et m) :

```
x <- 1
POUR i allant de 1 à n FAIRE
  x <- x+1
FIN POUR
POUR j allant de 1 à m FAIRE
  x <- x+1
FIN POUR
```

Exercice 3

Donner l'ordre de complexité des algorithmmes suivants (en fonction des variables n et m) :

```
x <- 1
POUR i allant n à n+6 FAIRE
  POUR j allant de m à m+3 FAIRE
      x  <- x+1
  FIN POUR
FIN POUR
```

Exercice 4

Donner l'ordre de complexité des algorithmmes suivants (en fonction des variables n et m) :

```
x <- 1
i <- 1
j <- 1

TANT QUE (i <=n) ET (j <=m) FAIRE

    x <- x+1
    i <- i+1
    j <- j+1
FIN TANT QUE

TANT QUE(i<=n) FAIRE
    x <- x+1
    i <- i+1
FIN TANT QUE

TANT QUE (j<=m) FAIRE
    x <- x+1
    j <- j+1
FIN TANT QUE
```

Les différents cas de complexité

Rappel du cours

- $O(1)$ temps constant
- $O(logn)$ logarithmique
- $O(n)$ linéaire
- $O(nlogn)$ quasi-linéaire
- $O(n^a)$ polynomial
- $O(2^n)$ exponentiel

Exemple

Par exemple, ici le programme cherche dans un tableau ordonné une valeur X demandée par l'utilisateur, dans un tableau de taille N, allant de 1 à 1000 inclus.

4	25	30	400	500

```
i <- 0
trouve <- 0
TANT QUE (i<N ET !trouve) FAIRE
    SI(monTab[i]=X) ALORS
      trouve <- 1
    FIN SI
    i <- i+1
FIN TANT QUE
```

- **Quel est le meilleur cas ?**
  - Le meilleur cas est lorsque l'on cherche la première valeur du tableau.
  - L'algorithme ne fera qu'un tour de boucle, et donc, peu d'opération/affectation/comparaison.

- **Quel est le pire cas ?**
  - Le pire cas est lorsque l'on cherche la dernière valeur du tableau ou une valeur qui n'y est pas !
  - L'algorithme fera N tours de boucle, et donc, plus d'opérations, affectations et/ou comparaisons.

- **Quel est le moyen cas ?**
  - C'est la situation moyenne, où l'algorithme met un temps moyen à s'exécuter. On suppose que les données sont réparties selon une certaines loi de probalités.
  - Considérons que l'on a 50 % de chance que x soit dans le tableau, et 50 %de chance qu'il n'y soit pas.
  - Sa position est au milieu
  - Le cas moyen est $O(N)$ soit N la taille du tableau

Exercices

Exercice 1

Nous considérons un jeu de lancer de pièce, qui permet de tirer au sort un vainqueur. Les règles sont les suivantes :

- S'il n'y a qu'un seul et unique participant, il est évidement le vainqueur
- S'il y a plus d'un participant, la pièce est lancée :
  - Si c'est face, on élimine au hasard un participant
  - Si c'est pile, on élimine au hasard deux participants
  - Le dernier participant restant est le vainqueur

Si l'on considère le lancer d'une pièce comme opération fondamentale, trouver l'équation de récurrence correspondant à ce jeu dans le cas moyen. Montrer par récurrence, à partir de cette équation, que l'algorithme précédent est en $\mathcal{O}(n)$, où n est le nombre initial de participants.

Exercice 2

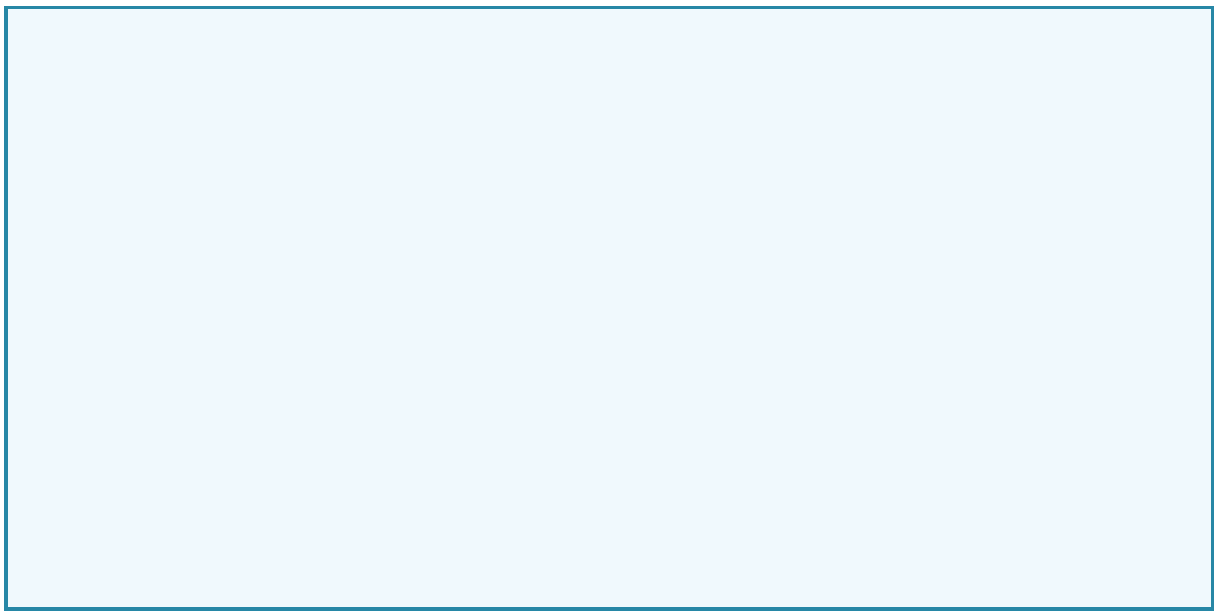
Nous considérons l'algorithme suivant :

```
1 s <- 0
2 POUR i allant de 1 à N FAIRE
3     SI ((T[i]*T[i] > 100) ALORS
4       s <- s+T[i]
5     FIN SI
6 FIN POUR
```

Les valeurs de T sont aléatoires entre 1 et 100 inclus

Quel est le nombre **d'additions** réalisées par cet algorithme dans :

1. Le meilleur cas
2. Le pire cas
3. Le cas moyen



Exercice 3

Nous considérons l'algorithme suivant, avec T est un tableau de booleen.

```
1 s <- 0
2 i <- 0
3 TANT QUE (i < n-1) FAIRE
4     SI (T[i] ET T[i+1]) ALORS
5       s <- s+1
6     FIN SI
7     i <- i+1
8 FIN TANT QUE
```

Exemple de tableau

i	0	1	2	3
T	Vrai	Faux	Faux	Vrai

1. Quels sont les tableaux qui représentent le meilleur et le pire cas de complexité en terme de nombre d'instructions effectuées ?
2. Calculer le nombre d'affectations, en supposant que chacun des booléens sont vrai ou faux, dans le pire cas, le meilleur cas et le moyen cas ?

1. Quels sont les tableaux qui représentent le meilleur et le pire cas de complexité ?

2. Calculer le nombre d'affectations, en supposant que chacun des booléens sont vrai ou faux, dans le pire cas, le meilleur cas et le moyen cas ?