

Complexité : le minimum

Prof d'info

Lycée Thiers

Complexité d'un programme

La *complexité en temps* mesure le nombre d'opérations effectuées par un programme en fonction de la taille des données. Dans ces transparents on étudie la *complexité temporelle au pire*.

On s'intéresse généralement à l'**ordre de grandeur** de ce coût.

Par exemple :

$$3n + 5 = O(n), \quad 2n^2 + 7n + 10 = O(n^2).$$

Les constantes multiplicatives sont ignorées, seul le terme tendant le plus vite vers l'infini est considéré.

Opérations élémentaires

On considère généralement que les opérations arithmétiques usuelles ont un coût constant :

$$+, \quad -, \quad \times, \quad /, \quad <, \quad ==, \dots$$

ont un coût

$$O(1).$$

Attention : cela ne signifie pas qu'une addition et une multiplication prennent exactement le même temps.

Cela signifie seulement que leur coût ne dépend pas de la taille n du problème étudié.

Suite d'instructions

```
1 x = x + 1;  
2 y = 2 * x;  
3 z = x + y;
```

Chaque instruction a un coût constant.

$$O(1) + O(1) + O(1) = O(1).$$

Plus généralement, pour une suite d'instructions :

$$T = T_1 + T_2 + \dots + T_k.$$

Instruction conditionnelle

```
1 if (x < y) {  
2     x = x + 1;  
3 }  
4 else {  
5     y = y + 1;  
6 }
```

On effectue :

- le test ;
- puis **une seule** des deux branches.

Ainsi, si les deux branches sont de coût constant, l'instruction complète est en

$$O(1).$$

Plus généralement, dans le pire cas :

$$\text{coût}(\text{test}) + \max(\text{coût}(\text{branche 1}), \text{coût}(\text{branche 2})).$$

Boucle

```
1 int i = 0;  
2 while (i < n) {  
3     x = x + i;  
4     i++;  
5 }
```

Le coût total de la boucle est la **somme des coûts** du corps de la boucle à chaque itération.

Si le corps a un coût C_i à l'itération i , alors :

$$T(n) = \sum_i C_i.$$

Dans cet exemple, le coût du corps est constant : $C_i = O(1)$.

Comme la boucle est exécutée n fois :

$$T(n) = \sum_{i=0}^{n-1} O(1) = O(n).$$

Coût d'une boucle : principe

Pour une boucle, on détermine :

- le nombre d'itérations ;
- le coût du corps à chaque itération.

Le coût total est :

$$\text{coût de la boucle} = \sum \text{coût du corps à chaque itération.}$$

On peut écrire

$$\text{nombre d'itérations} \times \text{coût d'une itération}$$

seulement lorsque le coût d'une itération est constant.

Boucles imbriquées

```
1 int i = 0, j = 0;
2 while (i < n) {
3     while (j < n) {
4         j = j + 1;
5     }
6     i = i + 1;
7 }
```

La boucle intérieure effectue n itérations. Son corps de boucle a un coût constant.

La boucle intérieure est elle-même exécutée n fois.

$$n \times n = n^2.$$

La complexité est donc

$$O(n^2).$$

Accès à un tableau

```
1 x = t[i];  
2 t[i] = 42;
```

Lorsque i est connu, accéder à la case $t[i]$ ou modifier cette case se fait en temps constant :

$$O(1).$$

Considérons :

```
1 int i = 0;  
2 // incrémenter de 1 les éléments du tableau  
3 while (i < n){  
4     t[i] = t[i] + 1;  
5     i = i + 1;  
6 }
```

Le corps de boucle consiste en un accès en lecture, une addition et un accès en écriture. Il y a n accès :

$$O(n).$$

Remarque sur les affichages

Le coût d'un appel à `printf` dépend de la quantité de texte affichée. Dans nos exemples, on pourra considérer l'affichage d'une valeur de type `int` ou `float` comme une opération de coût constant. En revanche, afficher une chaîne de caractères de longueur n a un coût en $O(n)$. Cette convention est une simplification : les entrées-sorties ont en réalité un coût plus difficile à modéliser précisément.

À retenir

Opération	Complexité
opération arithmétique	$O(1)$
comparaison	$O(1)$
accès à $t[i]$	$O(1)$
affectation à $t[i]$	$O(1)$
n opérations constantes	$O(n)$
deux boucles de taille n imbriquées	Au moins $O(n^2)$