O-notation

D.E ZEGOUR

École Supérieure d'Informatique

ESI

O-notation

Introduction

Le temps d ’exécution d'un programme dépend des facteurs suivants:

- les données du programme
- la qualité du code généré par le compilateur
- la machine (vitesse et nature des instructions)
- complexité de l'algorithme

En pratique, le temps moyen est souvent difficile à déterminer.

$T(n) = c\ f(n)$

n : nombre de données

c : constante regroupant les facteurs évoqués

f(n) : taux de croissance de T(n)

Essayer de trouver le cas moyen sinon trouver le cas le plus défavorable

Exemple : $T(n) = c\ n^2$

T(n) pourrait être le nombre d'instructions exécutées

O-notation

Définition

On dira que T(n) est O(f(n)) s'il existe $c > 0$ et $n0 > 0$ telles que $T(n) \leq c\, f(n)$ pour tout $n \geq n0$.

Un programme dont le temps d ’exécution est O(f(n)) est un programme qui a f(n) comme taux de croissance.

On dira aussi que f(n) est une limite supérieure du taux de croissance de T(n).

La constante dépend des facteurs liés à la machine et au code

O-notation

Exemple

$T(n) = O(n^2)$, cela veut dire
qu'il existe une constante $c > 0$ et une constante $n0 > 0$ tel que pour tout $n > n0$
$T(n) \leq c\ n^2$

Supposons que $T(0) = 1$, $T(1) = 4$ et en général
$T(n) = (n + 1)^2$
Montrons que $T(n)$ est $O(n^2)$

Il faut donc chercher une constante $c > 0$ et une constante $n0 > 0$ telles que
Quelque soit n : $n > 0$ on ait $T(n) <= c\ n^2$
C'est a dire $(n + 1)^2 <= cn^2$

O-notation

Exemple

$(n + 1)^2 \leq cn^2$

$n^2 + 2n + 1 <= c\ n^2$
$(c-1)\ n^2 - 2n - 1 \geq 0$

Donc pour c = 4 et n0 = 1 on a $T(n) \leq cn^2$ $T(n) = O(n^2)$
(ou $(c-1)n^2 - 2n - 1 \geq 0$

Delta = 4c

Pour c = 4, deux racines 1 et -1/3

O-notation

Ω-notation

Pour spécifier la limite inférieure du taux de croissance de T(n), on utilisera la notation $\Omega(g(n))$ qui veut dire: il existe une constante c > 0 telle que :

$T(n) \geq c\, g(n)$ pour un nombre infini de valeur de n.

Exemple : $T(n) = n^3 + 2n^2$ est $\Omega(n3)$ car pour c = 1 $T(n) \geq n^3$ pour n = 0, 1, 2, ...

O-notation

Opérations : règle de la somme

Si
T1(n) = O (f(n)) et
T2(n) = O(g(n))
sont les temps d'exécution de 2 fragments de programme P1 et P2,
Alors
le temps d'exécution de P1 suivi de P2 est
T1(n) + T2(n) = O (max (f(n), g(n))

Démonstration :

T1(n) = O(f(n) ⟶ Il existe c1 > 0 et n1 > 0 telles que quelque soit n > n1 : T1(n) ≤ C1 f(n)

T2(n) = O(g(n)) ⟶ Il existe c2 > 0 et n2 > 0 telles que quelque soit n > n2 : T2(n) ≤ C2 g(n)

T1(n) + T2(n) ≤ c1 f(n) + c2 g(n) pour un n0 = max (n1, n2)
≤ (c1 + c2) max (f(n), g(n))

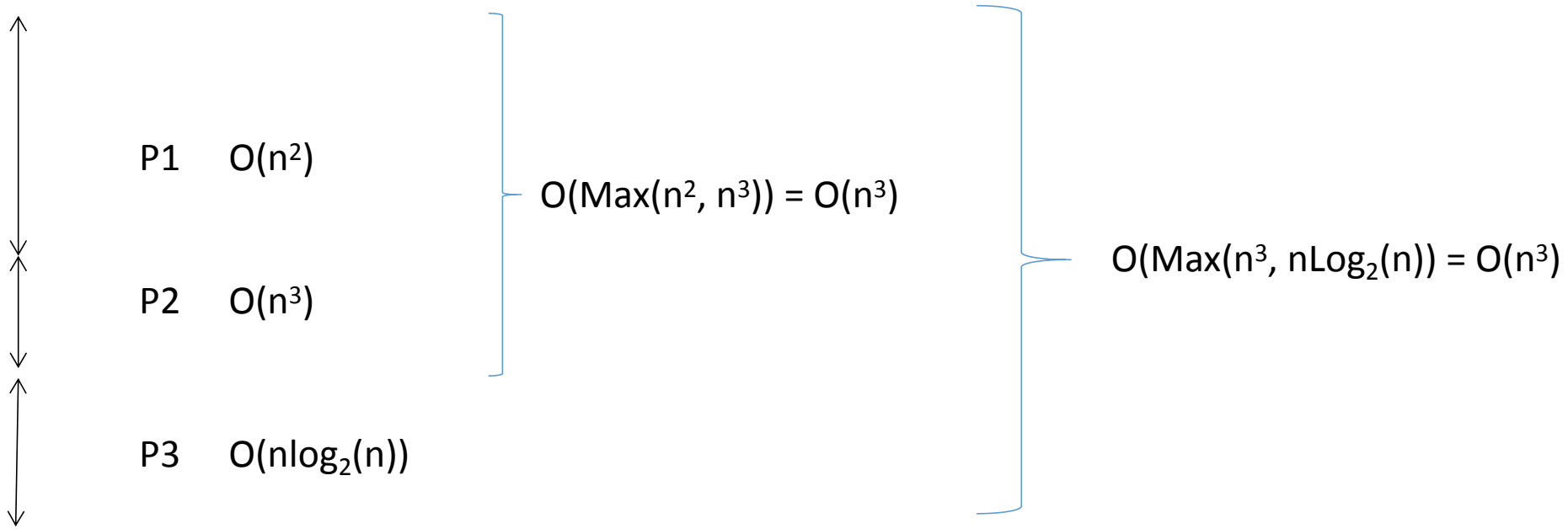
Donc il existe un c = c1 + c2 et n0 = max(n1,n2)

Conséquence : Si g(n) ≤ f(n) pour tout n > n0 alors (f(n) + g(n)) = O (f(n))
Exemple : $O(n^2 + n) = O(n^2)$

O-notation

Opérations : règle de la somme

-Utilisée pour calculer le temps d'exécution d'une séquence d ’étapes d'un programme.



O-notation

Opérations : règle du produit

Si T1(n) = O(f(n)) et
T2(n) = O(g(n))
Alors
T1(n) * T2(n) = O (f(n) * g(n))

Démonstration

T1(n) = O(f(n) ⟶ Il existe c1 > 0 et n1 > 0 telles que
quelque soit n > n1 : T1(n) ≤ C1 f(n)

T2(n) = O(g(n)) ⟶ Il existe c2 > 0 et n2 > 0 telles que
quelque soit n > n2 : T2(n) ≤ C2 g(n)

T1(n) * T2(n) ≤ c1 * c2 f(n) g(n) pour n > n0 avec n0 = max(n1, n2)

Il existe donc un c = c1 * c2 et un n0= max(n1, n2) tels que
T1(n) * T2(n) ≤ c f(n) g(n).

Donc T1(n)*T2(n)=O(f(n)g(n)).

Conséquence : O(c f(n)) = O(f(n)) si c > 0.
Exemple : $O(n^2/2) = O(n^2)$

O-notation

Mesure des algorithmes itératifs

1. Affectation, lecture ou écriture : O(1)

2. Séquence d'étapes : règle de la somme. C'est donc le temps de la séquence qui a le plus grand temps d ’exécution.

3. Alternative : se placer dans le cas le plus défavorble

4. Boucle : Règle du produit. C'est le produit du nombre d'itérations de la boucle par le plus grand temps possible pour une exécution du corps.

O-notation

Mesure des algorithmes itératifs : Exemple

```
(1) Pour i:= 1, n-1
(2)  Pour j := n,  i+1, -1
(3)    Si A(j-1) > A(j)
(4)         temps := A(j-1)
(5)         A(j-1) := A(j)
(6)         A(j) := temp
       Fsi
     Fpour
   Fpour
```

Tri par bulles d'un tableau A[1..n]

(4) (5) et (6) prennent chacun O(1)

Règle de la somme : (4) (5) et (6) est O(Max(1, 1, 1)) = O(1).

Pour l'instruction IF, O (Max(1, 1) = O(1)

Boucle (2) à (6) :corps : O(1) ;   boucle : O(n-i)
Règle du produit : O((n-i)*1) = O (n-i).

Boucle (1) (6): corps : O(n-i) ou O(n) (résultat précédent)
boucle : O(n-1) ou O(n)
Règle du produit : O(n.n) = $O(n^2)$