Module EA4 – Éléments d'Algorithmique

Dominique Poulalhon
dominique.poulalhon@liafa.univ-paris-diderot.fr

Université Paris Diderot
L2 Informatique, Math-Info et EIDD
Année universitaire 2013-2014

Points à retenir du cours précédent

algorithme = méthode systématique pour résoudre un problème

Points à retenir du cours précédent

algorithme = méthode systématique pour résoudre un problème

nécessite une preuve de correction : pour chaque entrée, l'algorithme doit terminer en produisant la bonne sortie

Points à retenir du cours précédent

algorithme = méthode systématique pour résoudre un problème

nécessite une preuve de correction : pour chaque entrée, l'algorithme doit terminer en produisant la bonne sortie

il peut exister plusieurs algorithmes pour le même problème

Points à retenir du cours précédent

algorithme = méthode systématique pour résoudre un problème

nécessite une preuve de correction : pour chaque entrée, l'algorithme doit terminer en produisant la bonne sortie

il peut exister plusieurs algorithmes pour le même problème

pour les comparer, il faut étudier leur complexité en temps et en espace

Complexité en espace

= quantité de mémoire nécessaire pour effectuer le calcul

Complexité en espace

= quantité de mémoire nécessaire pour effectuer le calcul

on ne tient compte que de la mémoire auxiliaire

i.e. on ne tient pas compte de la mémoire incompressible nécessaire pour stocker les données et le résultat

Complexité en temps

= temps nécessaire pour mener le calcul à son terme

Complexité en temps

= temps nécessaire pour mener le calcul à son terme

plus difficile à quantifier précisément, essentiellement car ce temps dépend de la machine utilisée

Complexité en temps

= temps nécessaire pour mener le calcul à son terme

plus difficile à quantifier précisément, essentiellement car ce temps dépend de la machine utilisée

convention : on exprime ce temps en nombre d'opérations élémentaires effectuées

opération élémentaire = opération dont le temps d'exécution peut être considéré comme constant

Complexité et ordres de grandeur

Nombre d'opérations effectuées par un ordinateur à 1 Ghz :

en 1 seconde	10^9
en 1 heure	$3 \cdot 10^{12}$
en 1 jour	$9 \cdot 10^{13}$
en 1 an	$3 \cdot 10^{16}$

Complexité et ordres de grandeur

n	10	100	10^3	10^6	10^9	10^{12}
$\log_2 n$	4	7	10	20	30	40
$10n$	100	10^3	10^4	10^7	10^{10}	10^{13}
$n \log_2 n$	34	665	10^4	$2 \cdot 10^7$	$3 \cdot 10^{10}$	$4 \cdot 10^{13}$
n^2	100	10^4	10^6	10^{12}	10^{18}	10^{24}
n^3	10^3	10^6	10^9	10^{18}	10^{27}	10^{36}
2^n	10^3	10^{30}	10^{301}	...	...	...

Complexité et ordres de grandeur

n	10	100	10^3	10^6	10^9	10^{12}
$\log_2 n$	4	7	10	20	30	40
$10n$	100	10^3	10^4	10^7	10^{10}	10^{13}
$n \log_2 n$	34	665	10^4	$2 \cdot 10^7$	$3 \cdot 10^{10}$	$4 \cdot 10^{13}$
n^2	100	10^4	10^6	10^{12}	10^{18}	10^{24}
n^3	10^3	10^6	10^9	10^{18}	10^{27}	10^{36}
2^n	10^3	10^{30}	10^{301}	...	...	...

Définition

Soit f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{N}$. On dit que :

- $f \in O(g)$, ou $f(n) \in O(g(n))$, ou $f(n) = O(g(n))$ si :

$$\exists c > 0,\ \exists n_0 \in \mathbb{N}, \quad \forall n \geqslant n_0,\ f(n) < c \cdot g(n)$$

- $f \in \Omega(g)$, ou $f(n) \in \Omega(g(n))$, si :

$$\exists c > 0,\ \exists n_0 \in \mathbb{N}, \quad \forall n \geqslant n_0,\ f(n) > c \cdot g(n)$$

- $f \in \Theta(g)$, ou $f(n) \in \Theta(g(n))$, si :

$$f(n) \in O(g(n)) \quad \text{et} \quad f(n) \in \Omega(g(n))$$

Calcul de la puissance n^E

```
def puissance(nb1, nb2) :
  res = 1
  for i in range(nb2) :
    res *= nb1
  return res
```

Calcul de la puissance n^E : l'exponentiation binaire

```
def puissance(nb1, nb2) :
  if nb2 == 0 : return 1
  tmp = puissance(nb1, nb2/2)
  carre = tmp * tmp
  if nb2%2 == 0 : return carre
  else : return nb1 * carre
```