Complexité

Luc Brun

luc.brun@greyc.ismra.fr

A partir de travaux de

Habib Abdulrab(Insa de Rouen)

Plan...

- Notion de complexité
- Comment évaluer la complexité d'un algorithme
- Exemple de calculs de complexité

Notion de complexité (1)

- Comment évaluer les performances d'un algorithme
- différents algorithmes ont des couts différents en termes de
  - temps d'exécution (nombre d'opérations effectuées par l'algorithme),
  - taille mémoire (taille nécessaire pour stocker les différentes structures de données pour l'exécution).

  Ces deux concepts sont appellé la complexité en temps et en espace de l'algorithme.

Notion de complexité (2)

- La complexité algorithmique permet de mesurer les performances d'un algorithme et de le comparer avec d'autres algortihmes réalisant les même fonctionnalités.
- La complexité algorithmique est un concept fondamental pour tout informaticien, elle permet de déterminer si un algorithme a et meilleur qu'un algortihme b et s'il est *optimal* ou s'il ne doit pas être utilisé...

Temps d'exécution (1)

Le temps d'exécution d'un programme dépend :

1. du nombre de données,
2. de la taille du code,
3. du type d'ordinateur utilisé (processeur,mémoire),
4. de la complexité en temps de l'algorithme «abstrait» sous-jacent.

Temps d'exécution (2)

Soit n la taille des données du problème et $T(n)$ le temps d'exécution de l'algorithme. On distingue :

- Le temps du plus mauvais cas $T_{max}(n)$ qui correspond au temps maximum pris par l'algorithme pour un problème de taille n.
- le temps moyen T_{moy} temps moyen d'exécution sur des données de taille n ($\Rightarrow$ suppositions sur la distribution des données).

Temps d'exécution

Règles générales :

1. le temps d'exécution (t.e.) d'une affectation ou d'un test est considéré comme constant c,
2. Le temps d'une séquence d'instructions est la somme des $t.e.$ des instructions qui la composent,
3. le temps d'un branchement conditionnel est égal au t.e. du test plus le max des deux t.e. correspondant aux deux alternatives (dans le cas d'un temps max).
4. Le temps d'une boucle est égal à la somme du coût du test + du corps de la boucle + test de sortie de boucle.

Problèmes du temps d'exécution (1)

- Soit l'algorithme suivant :

Nom: Calcul d'une somme de carrés
Role: Calculer la valeur moyenne d'un tableau
Entrée: n : entier
Sortie: somme : réel
Déclaration: i: Naturel
début
    somme ← 0.0
    pour i ←0 **à** n-1 **faire**
        somme ← somme+i*i
    finpour
fin

Problèmes du temps d'exécution (2)

$$\begin{aligned} T_{moy}(n) = T_{max}(n) &= c_1 + c_1 + n(c_2 + c_3 + c_4 + c_5 + c_1) \\ &= 2c_1 + n\left(\sum_{i=1}^{5} c_i\right) \end{aligned}$$

avec c_1 : affectation, c_2 : incrémentation, c_3 test, c_4 addition, c_5 multiplication.

Trop précis $\Rightarrow$
1. Faux,
2. Inutilisable.

Notion de complexité : comportement asymptotique du t.e.:

$$T_{max}(n) = T_{moy}(n) \approx nC$$

l'algorithme a une complexité en $\mathcal{O}(n)$ (ou $\mathcal{O}(Cn)$).

Complexité d'un algorithme récursif (1)

Soit l'algorithme :

```
fonction factorielle (n: Naturel) : Naturel
début
    si n = 0 alors
        retourner 1
    sinon
        retourner n*factorielle(n-1)
    finsi
fin
```

Complexité d'un algorithme récursif (2)

c_1 test, c_2 multiplication.

- T(0)=c_1
- T(n)=c_1+c_2+T(n-1)

$$\Rightarrow T(n) = nc_2 + (n+1)c_1$$

Complexité en $\mathcal{O}(n)$.

Les calculs de complexité d'algorithmes récursifs induisent natuellement des suites.

Principales complexités

- $\mathcal{O}(1)$: temps constant,
- $\mathcal{O}(\log(n))$: complexité logarithmique,
- $\mathcal{O}(n)$: complexité linaire,
- $\mathcal{O}(n\log(n))$
- $\mathcal{O}(n^2)$,$\mathcal{O}(n^3)$,$\mathcal{O}(n^p)$: polynomiales
- $\mathcal{O}(p^n)$ complexité exponentielle.

Influence de la complexité

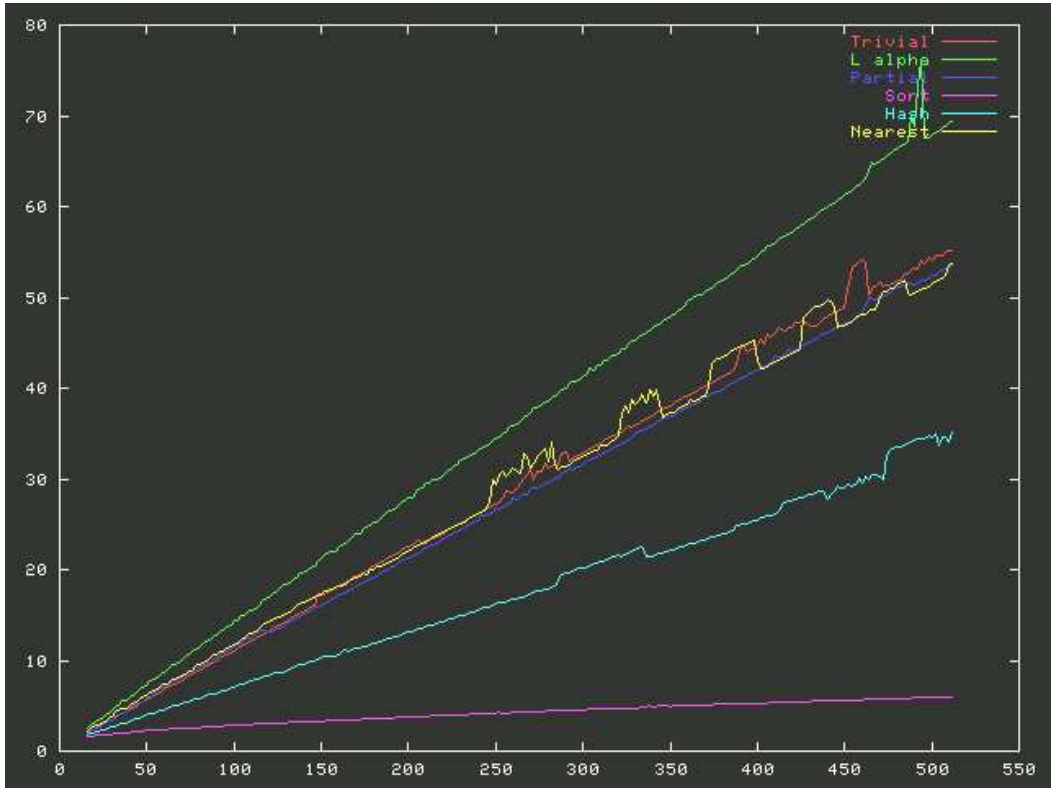
$n \to 2n$.

$\mathcal{O}(1)$	$\mathcal{O}(\log_2(n))$	$\mathcal{O}(n)$	$\mathcal{O}(n\log_2(n))$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(2^n)$
t	$t+1$	$2t$	$2t+2n$	$4t$	$8t$	t^2

	1	$\log_2(n)$	n	$n\log_2(n)$	n^2	n^3	2^n
$n=10^2$	$1\mu s$	$6\mu s$	$0.1ms$	$0.6ms$	$10ms$	$1s$	$4\times10^{16}a$
$n=10^3$	$1\mu s$	$10\mu s$	$1ms$	$10ms$	$1s$	$16.6min$	∞
$n=10^4$	$1\mu s$	$13\mu s$	$10ms$	$0.1s$	$100s$	$11,5j$	∞
$n=10^5$	$1\mu s$	$17\mu s$	$0.1s$	$1.6s$	$2.7h$	$32a$	∞
$n=10^6$	$1\mu s$	$20\mu s$	$1s$	$19.9s$	$11,5j$	32×10^3a	∞

Limites de la complexité

- La complexité est un résultat asymptotique : un algorithme en $\mathcal{O}(Cn^2)$ peut être plus efficace qu'un algorithme en $\mathcal{O}(C'n)$ pour de petites valeurs de n si $C \ll C'$,
- Les traitements ne sont pas toujours linéaires $\Rightarrow$ Il ne faut pas supposer d'ordres de grandeur entre les différentes constantes.



Conseils

Qualités d'un algorithme :

1. Maintenable (facile à comprendre, coder, déboguer),
2. Rapide

Conseils :

1. Privilégier le point 2 sur le point 1 uniquement si on gagne en complexité.
2. «ce que fait» l'algorithme doit se lire lors d'une lecture rapide : Une idée par ligne. Indenter le programme.
3. Faire également attention à la précision, la stabilité et la sécurité.

La rapidité d'un algorithme est un élément d'un tout définissant les qualités de celui-ci.