La complexité

Notion de complexité

Comment évaluer les performances d'un algorithme ?
Différents algorithmes ont des coûts différents en termes :

- de temps d'exécution (nombre d'opérations effectuées par l'algorithme),
- de taille mémoire (taille nécessaire pour stocker les différentes structures de données pour l'exécution).

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

La complexité algorithmique permet de mesurer les performances d'un algorithme et de le comparer avec d'autres algorithmes 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 algorithme *b* et s'il est optimal ou s'il ne doit pas être utilisé.

Temps d'exécution

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.

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 $T_{max}(n)$:
  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* (⇒ suppositions sur la distribution des données).

$$T_{moy}(n) = \sum_{i=1}^{r} p_i \, T_{si}(n)$$

  - *p_i probabilité que l'instruction s_i soit exécutée,*
  - *$T_{si}(n)$: temps requis pour l'exécution de s_i.*

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.

Principales complexités (avec la notation Grand-Θ)

- Θ(1) : temps constant,
- Θ(log(n)) : complexité logarithmique (Classe L),
- Θ(n) : complexité linaire (Classe P),
- Θ(n log(n)),
- $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(n^p)$: quadratique, cubique, polynomiale (Classe P),
- $\Theta(p^n)$: complexité exponentielle (Classe EXPTIME),
- Θ(n!) : complexité factorielle.

Limites de la complexité

La complexité est un résultat asymptotique : un algorithme en $\Theta(Cn^2)$ peut être plus efficace qu'un algorithme en $\Theta(C'n)$ pour de petites valeurs de n si $C \ll C'$.
Les traitements ne sont pas toujours linéaires. 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 :

- Privilégier le point 2 sur le point 1 uniquement si on gagne en complexité.
- «Ce que fait» l'algorithme doit se lire lors d'une lecture rapide : une idée par ligne, indenter le programme.
- 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.