Complexité des algorithmes: rappel des définitions de base

S'intéresser à la complexité (dynamique) d'un algorithme, c'est chercher à évaluer les ressources utilisées par l'algorithme. Plus précisément, on cherche à exprimer le coût de l'agorithme -la quantité de ressources utilisées- en fonction de la taille des données: les ressources considérées peuvent être le temps, la mémoire et le matériel (par exemple le nombre de processeurs, le nombre de canaux de communications ...); nous nous intéresserons la plupart du temps à la complexité temporelle. Avant de chercher donc à estimer la complexité, il faut préciser:

. La taille des données: Il faut d'abord définir ce qu'on entend par *taille des données*: c'est souvent le nombre de bits pour leur représentation en binaire mais ça peut être le nombre d'éléments pour une liste, la dimension pour une matrice, ...

. La notion de coût: On définit pour une donnée d, le coût de l'algorithme A pour cette donnée d: $cout_A(d)$. Nous nous placerons dans le cas des machines séquentielles "classiques" modélisées par la RAM. (Bien sûr, si on évalue la complexité d'un algorithme parallèle, le modèle choisi sera différent, par exemple PRAM.) On supposera de plus en général qu'une instruction élémentaire a un coût uniforme, i.e. indépendant de la taille de ses opérandes. (Par opposition au coût logarithmique où on tient compte de la taille des données). Enfin, on se borne en général à "compter" certaines instructions: par exemple, les comparaisons pour un tri par comparaisons, les accès à la mémoire externe pour un tri externe, les opérations arithmétiques de base pour des produits de matrices.

. Quelle(s) complexité(s) on cherche à calculer. En effet, pour deux données de même taille, l'exécution d'un algorithme peut utiliser des quantités de ressources différentes. On définit donc trois mesures de complexité, la complexité dans le meilleur des cas, la complexité en moyenne et la complexité dans le pire des cas:

- *La complexité dans le meilleur des cas:* elle est définie par
$$Inf_A(n) = inf\{cout_A(d)/d \ de \ taille \ n\}$$
- *La complexité dans le pire des cas:* elle est donnée par
$$Sup_A(n) = sup\{cout_A(d)/d \ de \ taille \ n\}$$
- *La complexité en moyenne:* pour la définir, il faut disposer pour tout n d'une mesure de probabilité p sur l'ensemble des données de taille n; cette mesure ne fait souvent qu'approximer l'espace réel des données, car il est très dur de proposer un modèle des données "réelles" -on suppose par exemple souvent que toutes les données de même taile sont équuiprobables, ce qui est peu réaliste-. La complexité en moyenne est alors donnée par
$$Moy_A(n) = \sum_{d \ de \ taille \ n} p(d) * cout(d)$$

Quand on parle de la complexité d'un algorithme sans préciser laquelle, c'est souvent de la complexité temporelle dans le pire des cas qu'on parle. La complexité dans le meilleur des cas n'est pas très utilisée; la complexité en moyenne est d'une certaine façon celle qui révèle le mieux le comportement "réel" de l'algorithme à condition de disposer d'un modèle de la répartition des données, mais bien sûr elle ne garantit rien sur le pire des cas ... et elle est souvent dure à calculer, même de façon approximative! Son calcul peut nécessiter la mise en œuvre de techniques mathématiques non élémentaires.

On ne calcule bien sûr en général pas exactement la complexité mais on se contente en général de calculer son ordre de grandeur voire de borner celui-ci. Par exemple, si on fait $n^2 + 2n - 5$ opérations, on retiendra juste que l'ordre de grandeur est n^2. On utilisera donc les notations classiques sur les ordres de grandeur:

Ordres de grandeur, Comportements asymptotiques:

Soient f et g deux fonctions de$N$ dans$R$:

. $f \in O(g)$ Ssi_{def} $\exists c \in R^{+*}, \exists A$,tels que $\forall n > A, f(n) \leq c * g(n)$

On dit que $f$est dominée asymptotiquement par g. On notera souvent $f = O(g)$

. $f \in \Theta(g)$ Ssi_{def} $\exists c, C \in R^{+*}, \exists A$,tels que $\forall n > A, C * g(n) \leq f(n) \leq c * g(n)$

On dit alors que f et g sont de même ordre de grandeur asymptotique. On notera souvent $f = \Theta(g)$.

On dira alors par exemple que la complexité d'un algorithme est "en $\Theta(n\ log\ n)$" ou en $O(n^2)$ ou ...

Vocabulaire: Un algorithme est dit

. en temps **constant** si sa complexité dans le pire des cas est bornée par une constante.

.**linéaire** (resp. linéairement borné) si sa complexité (dans le pire des cas) est en $\Theta(n)$ (resp. $O(n)$).

.**quadratique** (resp. au plus quadratique) si sa complexité (dans le pire des cas) est en $\Theta(n^2)$ (resp. en $O(n^2)$).

.**polynômial** ou polynômialement borné, si sa complexité (dans le pire des cas) elle est en $O(n^p)$ pour un certain p.

.**exponentiel** si elle est en $O(2^{n^p})$, pour un certain $p > 0$.

Remarque: On dit souvent simplement qu'un algorithme est linéaire si il est linéairement borné, i.e. si sa complexité (dans le pire des cas) est en $O(n)$, ou qu'il est quadratique si sa complexité dans le pire des cas est en $O(n^2)$. Dualement, on dit parfois qu'il est exponentiel si sa complexité est au moins exponentielle....

Algorithmes Praticables

Par "convention", un algorithme est dit **praticable** si il est polynômial; ceci appelle quelques remarques.

- Si un algorithme est exponentiel dans le pire des cas, il peut être polynomial en moyenne; si les pires cas sont exceptionnels, il peut être praticable ...en pratique: certains algos impraticables selon la définition ci-dessus sont ... pratiqués tous les jours (comme le simplexe).

+ Par contre, l'exécution d'un algorithme dont la complexité moyenne est de l'ordre de grandeur de n^5 microsecondes prendrait 30 ans si n=1000. Mais en pratique, il s'avère que la plupart des algorithmes polynômiaux ont un comportement asymptotique équivalent à un polynôme de faible degré, 2 ou 3.

+ De plus, la classe des algorithmes polynomiaux a de bonnes propriétés de clôture (par exemple, une "séquence" de deux algorithmes polynomiaux est polynomiale). Enfin, elle est indépendante du modèle séquentiel choisi: un algorithme polynômial pour le modèle des machines de Turing, le sera pour une RAM et vice-versa.

Le tableau ci-dessous montre le temps d'exécution de quatre algorithmes selon leur comportement et la taille des données; on suppose que la durée d'une instruction élémentaire est de l'ordre de la μs.

Taille / Comportement	$log_2 n$	n	n^2	2^n
10	$3\mu s$	$10\mu s$	$100\mu s$	$1000\mu s$
100	$7\mu s$	$100\mu s$	$1/100s$	10^{14} siècles
1000	$10\mu s$	$1000\mu s$	$1s$	astronomique
10000	$13\mu s$	$1/100s$	$1,7mn$	astronomique
100000	$17\mu s$	$1/10s$	$2,8h$	astronomique