Calculabilité et complexité

Traveaux pratiques 9

Complexité algorithmique

Y. Deville
C. Bertrand Van Ouytsel & V. Coppé & A. Gerniers & N. Golenvaux & M. Parmentier
Avril, 2021

1. - Pourquoi ne définit-on généralement pas la complexité d'un algorithme comme le nombre d'opération ou le temps d'exécution dans le *meilleur des cas* ?
   - Pourquoi ne définit-on généralement pas la complexité d'un algorithme comme la *moyenne* du nombre d'opération ou le temps d'exécution ?

   Pour les deux questions ci-dessus, donner au moins une raison théorique et une raison pratique pour justifier votre réponse.

2. Justifier rigoureusement les affirmations suivantes à partir de la définition de grand O :
   (a) $3n^3 + 2n^2 + n \in O(n^3)$
   (b) $\log_{10}(n^{100}) \in O(n)$
   (c) $n^4 \in O(3^n)$
   (d) $2^n \in O(n!)$

3. Supposons avoir deux algorithmes qui ont comme input et output des matrices carrées dont la taille n peut varier (la taille de la matrice en output n'est pas nécessairement la même que celle de la matrice en input). Supposons que la complexité du premier est dans $O(n^2)$ et que celle du second est dans $O(n^3)$.
   (a) Que peut-on dire sur la complexité de l'algorithme qui prend en entrée une matrice de taille n, applique le premier algorithme sur cette entrée, puis applique le second algorithme sur cette entrée, puis renvoie le résultat ainsi obtenu ?
   (b) Que peut-on dire sur la complexité de l'algorithme qui prend en entrée une matrice de taille n, applique le premier algorithme sur la matrice pour chaque nombre pair plus petit ou égal à n, puis applique le second algorithme sur la matrice pour chaque nombre impair plus petit ou égal à n, puis renvoie la somme des traces des résultats obtenus ?
   (c) Que peut-on dire sur la complexité de l'algorithme qui prend en entrée une matrice de taille n, applique le premier algorithme pour obtenir une seconde matrice, puis applique le second algorithme sur cette seconde matrice, puis renvoie le résultat ainsi obtenu ?

4. Est-il possible que le nombre d'opérations réalisées par un algorithme qui prend en entrée des listes de taille variable soit égal à 100 pour toute liste dont la taille est 42 si...

   (a) la complexité de l'algorithme est dans $O(2^n)$

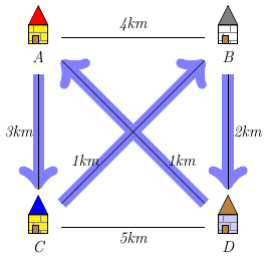
   (b) la complexité de l'algorithme est dans $\Omega(2^n)$

   (c) la complexité de l'algorithme est dans $\Theta(2^n)$

5. Le problème de "voyageur de commerce" ("*travelling salesman*" en anglais ; TSP) est un problème algorithmique classique. Un commerçant veut passer par chacune des n villes données. Il existe une route directe entre chaque paire de villes, chaque route ayant une certaine distance en km.

   Une instance du problème est donc représenté par un graphe avec n nœuds (les villes), qui est complet (une arrête entre chaque paire de nœuds) et pondéré (un poid (= distance) pour chaque arrête).

   Afin de minimiser ses coûts, notre commerçant souhaite trouver le circuit le plus court lui permettant de revenir à sa ville de départ après être passé une et une seule fois par chacune des autres villes.

   — Afin d'obtenir une solution exacte (pour toute entrée), on doit énumérer tous les circuits possibles visitant une et une seule fois chaque nœud du graphe, afin de sélectionner celui totalisant le moins de kilomètres. Quelle est la complexité d'une telle approche ?
   — Existe-t-il des cas pour lesquels on pourrait trouver la solution exacte en temps polynomial ? Si oui, donnez un exemple. Si non, justifiez.
   — Résoudre le TSP pour des grandes instances peut être intéressant pour l'industrie : e.g. un bras de robot qui resserre une centaine de boulons par voiture dans une ligne de production (minimiser la distance à parcourir peut augmenter la productivité de la ligne). Comment feriez-vous en pratique afin de définir le trajet du robot dans un temps raisonnable ?

Challenge

Supposons un instant que grâce à une technologie qui nous est gracieusement offerte par des extra-terrestres, nous sommes soudainement capables de toujours résoudre tout problème pour lequel il existe un algorithme dont la complexité (temporelle) est exponentielle en un temps polynomial. Cela implique-t-il que nous pourrons toujours résoudre tout problème en un temps polynomial ?