Rappels de complexité algorithmique

Arnaud Casteigts

Bachelor en sciences informatiques,
Université de Genève

Complexité algorithmique ?

Ressources nécessaires pour résoudre un problème.

Quel type de ressources ?

- **Temps** (nombre d'opérations)
- Espace (quantité de mémoire)
- ...

Les valeurs exactes dépendent du modèle de machine considéré. On en parlera peu.

Point de vue **asymptotique**

- **Croissance** de ces quantités en fonction de la **taille** n de l'entrée, quand $n \to \infty$
- Notations $O(\cdot), \Omega(\cdot), \Theta(\cdot)$ (ignore les **facteurs constants** et les **termes dominés**)
  Intuition $\leq$ $\geq$ $=$ Ex : $3n^2 + 5n + 4 = \Theta(n^2)$
- Quelques adjectifs (ici pour $O(\cdot)$) :

Constant	$O(1)$	Quadratique	$O(n^2)$	
Logarithmique	$O(\log n)$	Exponentiel	$O(2^n)$	
Linéaire	$O(n)$	Factoriel	$O(n!)$	
Quasi-linéaire	$O(n \log n)$	Polynomial	$O(n^c) = n^{O(1)}$	$\approx$ rapide

En général, on s'intéresse au **pire cas**, ou parfois au **cas moyen** (p.ex. sur des instances aléatoires).
Dans les deux cas, on se contentera souvent dans ce cours d'utiliser la notation $O()$.

Exercice 1 : ordres de grandeur

	True	False
$4n^2 - 5n + 1 = O(n^2)$	✓	
$4n^2 - 5n + 1 = \Theta(n^2)$	✓	
$n \log n = \Omega(n)$	✓	
$n \log n = O(n^2)$	✓	
$n \log n = \Theta($ __________ $)$?	N/A	N/A
$n \log n^2 = \Theta($ __________ $)$?	N/A	N/A
$500 = \Theta(1)$	✓	
$17n^2 + 3 = O(n^{\Theta(1)})$	✓	
$\sqrt{n} = O(n)$	✓	
$n! = \Omega(2^n)$	✓	

Exercice 2 : complexité de quelques problèmes

Quelle est la complexité en temps de ces problèmes (dans le pire cas) ?

1. Décider si une liste de taille n contient un élément donné ? $O(n)$
2. Décider si une liste triée de taille n contient un élément donné ? $O(\log n)$
3. Vérifier si une liste est triée ? $O(n)$
4. Trier une liste ? $O(n \log n)$
5. Décider si une liste contient des doublons ? naïf : $O(n^2)$, mieux : $O(n \log n)$
6. Parcourir la matrice d'adjacence d'un graphe à n sommets ? $O(n^2)$
7. Énumérer tous les sous-ensembles d'un ensemble de taille n ? $O(2^n)$
8. Idem, en énumérant seulement les sous-ensembles de taille 3 ? $O(n^3)$
9. Énumérer tous les ordres de visite possibles entre n villes ? $O(n!)$
10. Décider si un graphe à n sommets contient une clique de taille 17 $O(n^{17})$
11. Décider si un graphe donné est 3-colorable ? naïf : $O(3^n)$, mieux : $O(1.3289^n)$

Types de problèmes

- **Décision :** $\{0,1\}^* \to \{0,1\}$
  Ex : est-ce que la photo représente un chat ?
  Ex : existe-t-il un chemin de A vers B ?
- **Recherche :** $\{0,1\}^* \to \{0,1\}^*$
  Ex : trouver un chat
  Ex : trouver un chemin quelconque de A vers B

- **Dénombrement (comptage) :** $\{0,1\}^* \to \mathbb{N}$
  Ex : combien y-a-t'il de chats ?
  Ex : combien y-a-t'il de chemins de A vers B
- **Optimisation :** $\{0,1\}^* \to \{0,1\}^*$
  Ex : trouver le chat le plus mignon.
  Ex : trouver le plus court chemin de A vers B

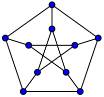
Principales classes de problèmes (de décision)

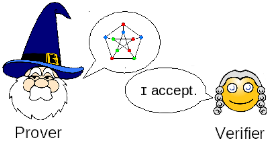
- P : problèmes **résolubles** en temps polynomial.
- NP : problèmes **vérifiables** en temps polynomial.
  $\rightarrow$ Si la réponse est OUI, il existe une **preuve** de cela qui peut être vérifiée rapidement

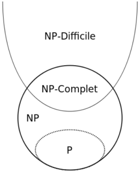
Exemple : 3-COLORING

Un graphe donné peut-il être colorié avec 3 couleurs ?

$\rightarrow$ difficile à résoudre
$\rightarrow$ mais facile à vérifier







- NP-difficile (NP-hard) : au moins aussi difficile à **résoudre** que tout problème de NP
- NP-complet : à la fois NP-difficile et dans NP

Exemples : 3-COLORING, CLIQUE, TSP, SAT, SET COVER...
et de nombreux autres dont on reparlera.

La grande question : est-ce que $\mathrm{P} = \mathrm{NP}$? (on n'en parlera pas ici)

Dans ce cours, on gardera en tête qu'on ne sait pas résoudre un problème NP-complet rapidement et dans tous les cas. On le fera quand même, lentement ou dans des cas spécifiques.

On s'intéressera aussi à de nombreux problèmes de faible complexité.