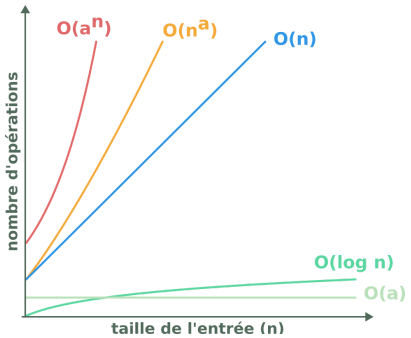
Qu'est-ce qu'un algorithme efficace ?

Faculté des Sciences, Département d'Informatique
Abdalrahman EL HUSSEIN, Sam NADIF, Clément SINON et Alexis SMEETS

INTRODUCTION

Un **algorithme** est une suite finie d'instructions précises à suivre pour résoudre un problème. L'algorithmique — l'étude des algorithmes — est une branche des mathématiques particulièrement étudiée en informatique car un programme est simplement la mise en œuvre dans un algorithme en langage de programmation.

Le nombre d'instructions à suivre peut varier en fonction de la taille des données du problème, et pas nécessairement de manière linéaire. Avec un tri « naïf »[1], trier une liste de 20 éléments nécessite quatre fois plus d'étapes que de trier une liste de 10 éléments. On appelle **complexité algorithmique** la façon dont le nombre d'opérations augmente en fonction de la taille des données. Lorsqu'on parle d'**efficacité d'un algorithmique**, on fait généralement référence à sa complexité[2].



Quelques ordres de complexité (avec *a* une constante quelconque)

La notation grand-O est utilisée pour décrire la complexité. « O(f(n)) » signifie que, pour des données de grande taille, le nombre d'opérations de l'algorithme ne croît pas plus vite que f(n), à un

OBJECTIFS D'APPRENTISSAGE

1. Définir ce qu'est un algorithme et le différencier d'un programme.
2. Comprendre qu'une complexité exponentielle peut vite croître, au point d'être infaisable pour un ordinateur.
3. Comprendre qu'une différence de complexité peut avoir un énorme impact sur de grandes données.

---

[1] Un tri de complexité quadratique (n^2). Il existe des tris de complexité linéarithmique ($n \cdot \log n$).
[2] On considère souvent qu'un algorithme est efficace si sa complexité est polynomiale ou moindre, par opposition à des complexités exponentielles.

ACTIVITÉS

Notre stand s'appuie sur **le problème du rendu de monnaie** pour introduire par l'exemple le concept de complexité : *« Si je dois rendre 67 ¢ à un client, quelle combinaison minimale de pièces et de billets puis-je lui rendre ? ».*

1. À l'aide d'une grille à remplir, réaliser l'algorithme glouton : prendre la plus grosse pièce autant de fois que possible jusqu'à atteindre la somme recherchée.
2. Découvrir l'algorithme naïf : considérer toutes les combinaisons possibles de pièces jusqu'à en trouver une qui forme la somme demandée.
3. Formuler la complexité des algorithmes glouton et naïf : O(C) et $O(C^V)$ respectivement, avec C le nombre de pièces/billets différents disponible et V la valeur de la somme à rendre.
4. Course entre un ordinateur moderne exécutant l'algorithme naïf et un vieil ordinateur des années '80 exécutant l'algorithme glouton, afin de démontrer le caractère explosif d'une complexité exponentielle.
5. *(bonus)* À l'aide d'une grille à remplir, réaliser l'algorithme de programmation dynamique (DP) : une amélioration de l'algorithme naïf avec une complexité O(C·V).

EXEMPLES DE QUESTIONS

Qu'est-ce qu'un algorithme ?
Réponse : Une suite d'instructions permettant de résoudre un problème.
Piège : Ne pas dire « une suite d'instructions exécutée par un ordinateur » ; un algorithme peut tout aussi bien être exécuté par un humain.

Le flouze est une monnaie avec C = 4 pièces différentes. Je dois rendre V = 3 flouzes à un client. Jusqu'à combien de calculs vais-je devoir effectuer pour savoir quoi lui rendre selon :
- l'algorithme glouton, de complexité O(C) : 4
- l'algorithme naïf, de complexité $O(C^V)$: $4^3 = 64$
- (bonus) l'algorithme DP, de complexité O(C·V) : $4 \cdot 3 = 12$
Remarque : Il n'est pas pertinent de mémoriser les complexités exactes de ces trois algorithmes ; l'important est de comprendre comment elles évoluent lorsque les paramètres augmentent.

Replacer les algorithmes naïf, glouton et DP (bonus) sur le graphe des complexités.