المدرسة الحسنية للأشغال العمومية
ECOLE HASSANIA DES TRAVAUX PUBLICS

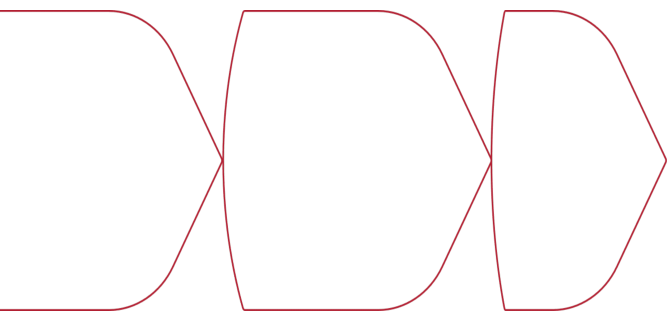
Introduction à la Complexité

Comprendre les bases de la théorie de la complexité

Slimani Mohamed Amine

EHTP

February 11, 2025



Sommaire

Qu'est-ce que la complexité ?

- **Définition** : La complexité est une mesure des ressources nécessaires pour résoudre un problème, souvent en termes de temps et d'espace.
- **Objectif** : Évaluer l'efficacité des algorithmes et comprendre les limites de la calculabilité.
- **Avantages** : Permet de comparer des algorithmes et de prédire leur comportement sur de grandes entrées.

Pourquoi étudier la complexité ?

- **Optimisation** : Identifier les algorithmes les plus efficaces pour un problème donné.
- **Prédiction** : Anticiper les performances des algorithmes sur de grandes entrées.
- **Théorie** : Comprendre les limites fondamentales de ce qui peut être calculé.

Concepts de base de la complexité

- **Complexité en temps** : Nombre d'étapes nécessaires pour résoudre un problème.
- **Complexité en espace** : Quantité de mémoire nécessaire pour résoudre un problème.
- **Notation Big-O** : Utilisée pour décrire la croissance asymptotique d'une fonction.

Classes de complexité

O	**Type de complexité**
$O(1)$	Constante
$O(\log(n))$	Logarithmique
$O(n)$	Linéaire
$O(n \times \log(n))$	Quasi-linéaire
$O(n^2)$	Quadratique
$O(n^3)$	Cubique
$O(2^n)$	Exponentielle
$O(n!)$	Factorielle

Exemple d'analyse de complexité

Analyse de complexité

```
# Exemple de recherche linéaire
Comment Code
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1
# Complexité en temps : O(n)
```

Bonnes pratiques

- **Choix d'algorithmes** : Préférer les algorithmes avec une complexité en temps et espace optimale.
- **Analyse** : Toujours analyser la complexité avant de choisir un algorithme.
- **Optimisation** : Éviter les optimisations prématurées sans comprendre la complexité.

Outils pour analyser la complexité

- **Profiling** : Utiliser des outils de profiling pour mesurer les performances.
- **Simulation** : Simuler des entrées de grande taille pour prédire le comportement.
- **Théorie** : Étudier la théorie de la complexité pour mieux comprendre les limites.

Exemple d'application avec Python

Analyse de complexité en Python

```python
# Exemple de tri par sélection
Comment Code
def selection_sort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i+1, len(arr)):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
# Complexité en temps : O(n^2)
```

Défis de la complexité

- **Problèmes NP-Complets** : Résoudre ces problèmes de manière efficace reste un défi.
- **Grandes entrées** : Les algorithmes peuvent devenir inefficaces sur de très grandes entrées.
- **Optimisation** : Trouver un équilibre entre temps et espace peut être difficile.

Pourquoi c'est important ?

- La complexité est essentielle pour comprendre les limites des algorithmes.
- Elle permet de choisir les meilleurs algorithmes pour des problèmes spécifiques.
- Comprendre la complexité est crucial pour les développeurs et les chercheurs en informatique.

Résumé

La complexité est un concept fondamental en informatique pour comprendre et optimiser les algorithmes. Explorez, apprenez, et innovez avec la théorie de la complexité !