Algorithme et complexité

Ambre Le Berre

MPSI

2025/2026

Plan du cours

1. Programme vs Algorithme
2. Complexité d'un algorithme

Programme vs Algorithme

Un algorithme ?

En programmation, on parle souvent d'algorithme. C'est une sorte de programme abstrait.

Un algorithme ?

En programmation, on parle souvent d'algorithme. C'est une sorte de programme abstrait.

Recherche linéaire(L, x)

Entrée : L une liste, x un élément.

Sortie : true si x est dans le liste, false sinon.

```
1  Pour chaque élément y dans la liste L
2  Si x = y
3      Renvoyer true
4  Renvoyer faux
```

Implémentation d'un algorithme

Une implémentation d'un algorithme est une traduction de l'algorithme dans un language de programmation. Cela peut impliquer une réfléxion et des choix supplémentaires.

Implémentation d'un algorithme

Une implémentation d'un algorithme est une traduction de l'algorithme dans un language de programmation. Cela peut impliquer une réfléxion et des choix supplémentaires.

Écrivez une implémentation en Python de l'algorithme de recherche linéaire.

Correction

```
def recherche_lineaire(L, x):
    """Cherche un élément x dans une liste L. Renvoie True si
l'élément est présent, False sinon."""
    for y in L:
        if x = = y:
            return True
    return False
```

Complexité d’un algorithme

Introduction

La complexité d'un algorithme est une estimation du temps qu'il faut pour qu'il s'execute.

Introduction

La complexité d'un algorithme est une estimation du temps qu'il faut pour qu'il s'execute.

Définition 2 [Opération élémentaire]

> On appelle opération élémentaire toute opération "de base" d'un language : addition, multiplication, division, comparaisons, modulo ...

On va compter les opérations élémentaires dans un programme ou un algorithme.

Exemple

Recherche linéaire(L, x)

Entrée : L une liste, x un élément.

Sortie : true si x est dans le liste, false sinon.

```
1  Pour chaque élément y dans la liste L
2  Si x = y
3      Renvoyer true
4  Renvoyer faux
```

Exemple

Recherche linéaire(L, x)

Entrée : L une liste, x un élément.

Sortie : true si x est dans le liste, false sinon.

```
1  Pour chaque élément y dans la liste L
2  Si x = y
3      Renvoyer true
4  Renvoyer faux
```

Est ce qu'on compte la boucle elle-même ? Et les retours ?

Comparaison des opérations

Opération	Cycles d'execution
Addition, soustraction	1
Multiplication	5
If	1 - 20
Division / Modulo	30
Accès mémoire	1 - 150
Lecture sur le disque	~10000

Est ce que ça a du sens de compter toutes les opérations une par une ?

Complexité asymptotique

On va seulement prendre l'ordre de grandeur du nombre d'opérations élémentaire, en fonction de la taille de l'entrée (en général nommée n).

On utilise la notation $O(\ldots)$, pour dire "au plus de l'ordre de ..."

Exemples

On peut avoir par exemple :

- $O(1)$: signifie que le nombre d'opérations **ne dépend pas** de la taille de l'entrée.
- $O(n)$: signifie que le nombre d'opérations est proportionnel à la taille de l'entrée.
- $O(n^2)$: signifie que le nombre d'opérations est proportionnel au carré de la taille de l'entrée.

...

Exemples

On peut avoir par exemple :

- $O(1)$: signifie que le nombre d'opérations **ne dépend pas** de la taille de l'entrée.
- $O(n)$: signifie que le nombre d'opérations est proportionnel à la taille de l'entrée.
- $O(n^2)$: signifie que le nombre d'opérations est proportionnel au carré de la taille de l'entrée.

...

Laquelle de ces options correspond à l'algorithme de recherche linéaire ?

Définition formelle

Définition 3 [Notation $O()$]

On dit qu'une fonction f est un "grand O" d'une fonction g lorsque, si

$\exists C \in \mathbb{N}, N_0 \in \mathbb{N}$ tels que $\forall n \geq N_0,\ f(n) < C \cdot g(n)$

Soit : Il existe une constante C, telle que, pour n suffisemment grand, $g(n)$ soit majorée par $C \cdot g(n)$

Exemples :

- $5n + 3$ est un $O(\ldots)$
- $6n^2 + n + 100000000$ est un $O(\ldots)$
- $\frac{n}{n+1}$ est un $O(\ldots)$
- $\frac{3n^2-5n+20}{3n-1}$ est un $O(\ldots)$

Exemples :

- $5n + 3$ est un $O(n)$
- $6n^2 + n + 100000000$ est un $O(n^2)$
- $\frac{n}{n+1}$ est un $O(1)$
- $\frac{3n^2-5n+20}{3n-1}$ est un $O(n)$

Comment calculer la complexité d'un algorithme ?

Dans votre cas :

1. Sauf exception, toutes les opérations et fonctions fournies sont en temps constant, soit `O(1)`.
2. Cas des boucles : Si une boucle fait n itérations, et que chaque itération est, dans le pire cas, en $O(p)$, alors le total est en $O(np)$

Exemple

```
def recherche_lineaire(L, x):
    """Cherche un élément x dans une liste L. Renvoie True si
l'élément est présent, False sinon."""
    for y in L:
        if x == y:
            return True
    return False
```

Exemple

```
def recherche_lineaire(L, x):
    """Cherche un élément x dans une liste L. Renvoie True si
l'élément est présent, False sinon."""
    for y in L:
        if x == y:
            return True
    return False
```

On a `len(L)` itéations de la boucle, et chaque itération est en $O(1)$. Donc $O(\text{len(L)})$ au total.