INTRODUCTION À L'INFORMATIQUE CM3

Antonio E. Porreca
https://aeporreca.org/teaching

DÉCRIRE DES ALGORITHMES

- En langage naturel (e.g., en français)
- En pseudocode (semi-formel)
- En langage de programmation (formel)
    - E.g. en Python dans l'UE Mise en œuvre informatique

RECHERCHE DANS UNE SÉQUENCE EN LANGAGE NATUREL

- Pour chaque élément de la séquence à partir du premier :
    - Si cet élément est l'élément cherché, on a terminé
    - Sinon, on continue avec l'élément suivant
- S'il n'y a plus d'éléments et on n'a pas trouvé ce qu'on cherchait, alors il n'est pas là

RECHERCHE DANS UNE SÉQUENCE EN PSEUDOCODE

```
fonction chercher(élément, séquence)
    n := longueur(séquence)
    i := 0
    tant que i < n faire
        si séquence[i] = élément alors
            retourner i
        fin
        i := i + 1
    fin
    retourner -1
fin
```

RECHERCHE DANS UNE SÉQUENCE EN PYTHON

```
def chercher(element, sequence):
    n = len(sequence)
    i = 0
    while i < n:
        if sequence[i] == element:
            return i
        i = i + 1
    return -1
```

INDENTATION DU (PSEUDO)CODE

fonction chercher(élément, séquence)
n ≔ longueur(séquence)
i ≔ 0
tant que i < n **faire**
si séquence[i] = élément **alors**
retourner i
fin
i ≔ i + 1
fin
retourner −1
fin

INDENTATION DU (PSEUDO)CODE

```
fonction chercher(élément, séquence)
    n ≔ longueur(séquence)
    i ≔ 0
    tant que i < n faire
        si séquence[i] = élément alors
            retourner i
        fin
        i ≔ i + 1
    fin
    retourner −1
fin
```

EXÉCUTION D'UN ALGORITHME ET AFFECTATIONS DES VARIABLES

```
fonction chercher(élément, séquence)
    n := longueur(séquence)
    i := 0
    tant que i < n faire
        si séquence[i] = élément alors
            retourner i
        i := i + 1
    retourner -1
```

STRUCTURES DE CONTRÔLE

instruction$_1$
instruction$_2$
...
instruction$_n$

si *condition* **alors**
    instructions
sinon
    d'autres instructions
[**fin**]

tant que *condition* **faire**
    instructions
[**fin**]

SÉQUENCE D'INSTRUCTIONS

fonction discriminant(a, b, c)
(*calcul du* **Δ** *de l'équation* $ax^2 + bx + c = 0$)
$d \coloneqq b^2$
$e \coloneqq 4ac$
$\Delta \coloneqq d - e$
retourner Δ

CONDITIONS

fonction solutions(a, b, c)
    (*calcul des solutions de* $ax^2 + bx + c = 0$)
    $\Delta \coloneqq b^2 - 4ac$ (*discriminant*)
    si $\Delta > 0$ **alors**
        $x_1 \coloneqq (-b + \sqrt{\Delta}) / 2a$
        $x_2 \coloneqq (-b - \sqrt{\Delta}) / 2a$
        retourner $\{x_1, x_2\}$ (*deux solutions*)
    sinon si $\Delta = 0$ **alors**
        retourner $\{-b / 2a\}$ (*une solution*)
    sinon
        retourner $\varnothing$ (*aucune solution*)

ITÉRATION

```
fonction puissance(a, n)
    (calcul de la puissance a^n avec n ≥ 0)
    si n < 0 alors
        erreur « exposant négatif »
    p := 1
    m := n
    tant que m > 0 faire
        p := p x a
        m := m – 1
    retourner p
```

COMPOSITION DES STRUCTURES DE CONTRÔLE

```
si x < 10 alors
    si y < 30 alors
        x = x + y
    sinon
        y = y – x
sinon
    tant que x < 20 faire
        tant que y > 40 faire
            y := y – 1
            x := x + 1
```

APPELER UNE FONCTION

fonction solutions(a, b, c)
    (*calcul des solutions de* $ax^2 + bx + c = 0$)
    Δ := discriminant(a, b, c)
    si $\Delta > 0$ **alors**
        $x_1 := (-b + \sqrt{\Delta}) / 2a$
        $x_2 := (-b - \sqrt{\Delta}) / 2a$
        retourner $\{x_1, x_2\}$
    sinon si $\Delta = 0$ **alors**
        retourner $\{-b / 2a\}$
    sinon
        retourner $\varnothing$

APPELER UNE FONCTION

fonction solutions(a, b, c)
(*calcul des solutions de* $ax^2 + bx + c = 0$)
Δ := discriminant(a, b, c)
si $\Delta > 0$ **alors**
$x_1 := (-b + \sqrt{\Delta}) / 2a$
$x_2 := (-b - \sqrt{\Delta}) / 2a$
retourner $\{x_1, x_2\}$
sinon si $\Delta = 0$ **alors**
retourner $\{-b / 2a\}$
sinon
retourner $\varnothing$

fonction discriminant(a, b, c)
$d := b^2$
$e := 4ac$
$\Delta := d - e$
retourner Δ

TYPES DE DONNÉES

fonction puissance(a : réel, n : entier) : réel
(*calcul de la puissance* a^n *avec* $n \geq 0$)
si $n < 0$ **alors**
erreur « exposant négatif »
p : réel := 1
m : entier := n
tant que $m > 0$ **faire**
$p := p \times a$
$m := m - 1$
retourner p

PROPRIÉTÉS SOUHAITÉES POUR UN ALGORITHME

- Terminaison : l'algorithme termine en un temps fini
- Correction : l'algorithme calcule le bon résultat
- Efficacité : l'algorithme est aussi rapide que possible (ou au moins aussi rapide que nécessaire)

SOMME DES N PREMIERS ENTIERS

```
fonction somme(n)
    s ≔ 0
    i ≔ 1
    tant que i ≤ n faire
        s ≔ s + i
        i ≔ i + 1
retourner s
```

Terminaison ?

Correction ?

Efficacité ?

SOMME DES N PREMIERS ENTIERS

```
fonction somme(n)
    s ≔ 0
    i ≔ 1
    tant que i ≤ n faire
        s ≔ s + i
        i ≔ i + 1
retourner s
```

```
fonction somme-à-la-Gauss(n)
    retourner n (n + 1) / 2
```

Terminaison ?

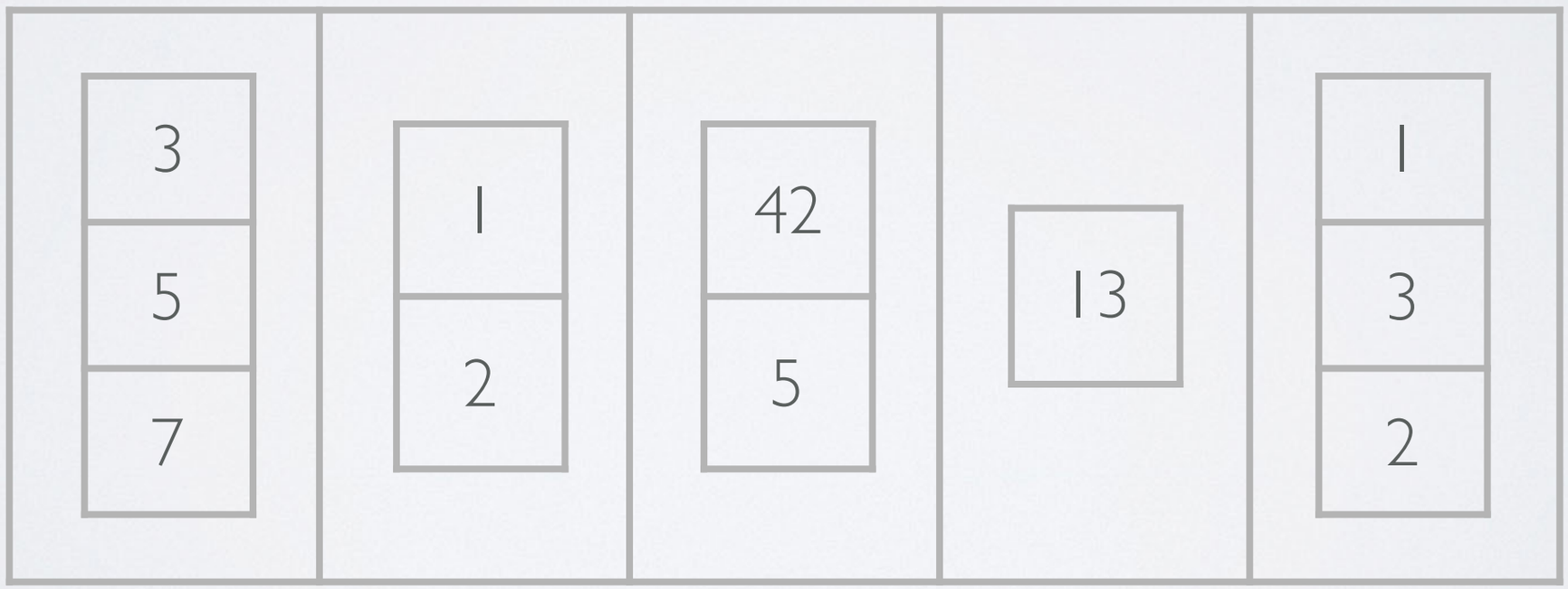
Correction ?

Efficacité ?

STRUCTURES DE DONNÉES : LES TABLEAUX

STRUCTURES DE DONNÉES : LES TABLEAUX

TABLEAUX



PARCOURIR UN TABLEAU

```
procedure parcours(T)
    n ≔ longueur(T)
    i ≔ 0
    tant que i < n faire
        écrire T[i]
```

RECHERCHE DANS UN TABLEAU

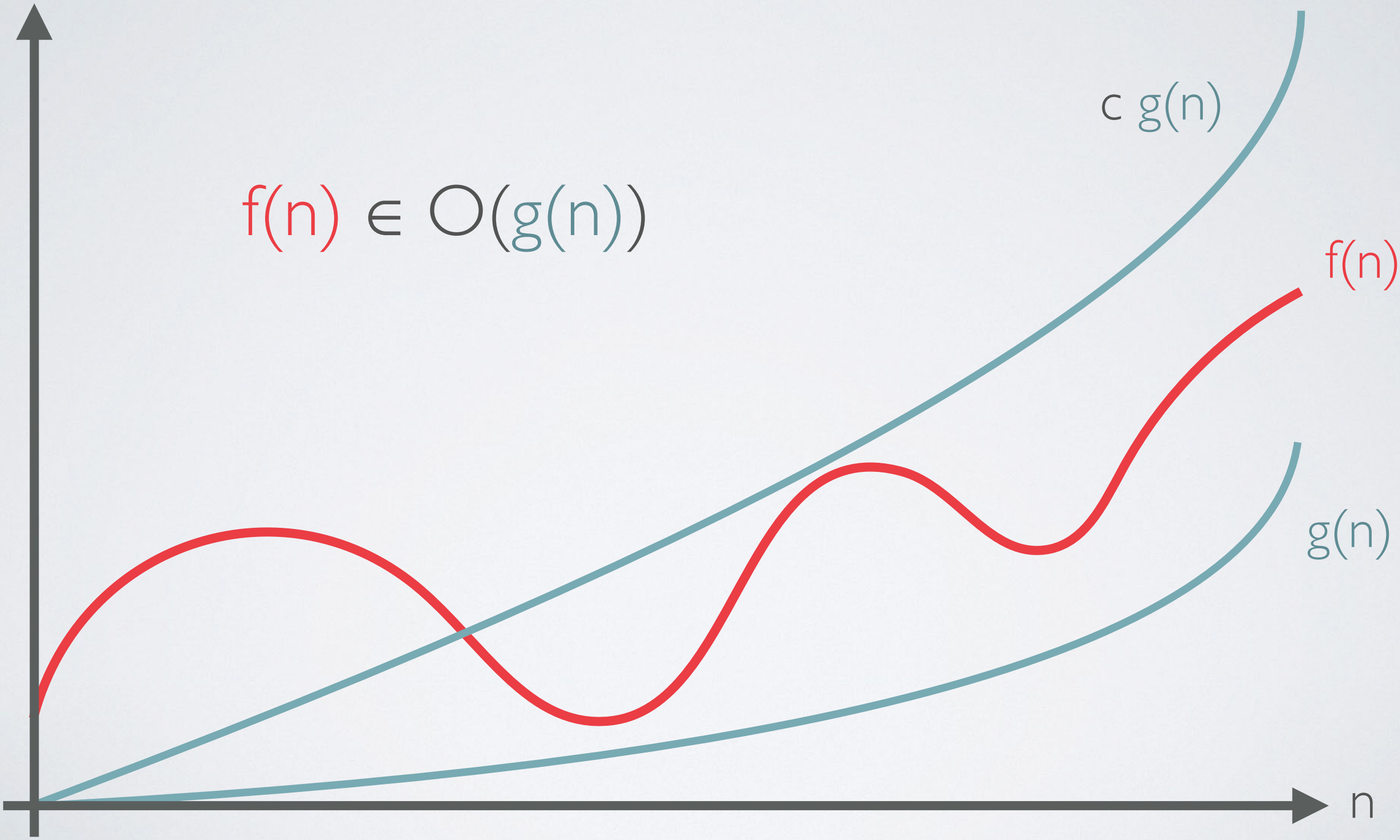
```
fonction chercher(x, T)
    n ≔ longueur(T)
    i ≔ 0
    tant que i < n faire
        si T[i] = x alors
            retourner i
        i ≔ i + 1
    retourner −1
```

Terminaison ?

Correction ?

Efficacité ?

NOTATION « GRAND O »



ORDRES DE GRANDEUR

- $n \in O(n)$
- $n + 5 \in O(n)$
- $2n + 5 \in O(n)$
- $n^2 \notin O(n)$

RECHERCHE DANS UN ANNUAIRE OU UN DICTIONNAIRE ?

RECHERCHE DICHOTOMIQUE DANS UN TABLEAU D'ENTIERS TRIÉ

```
fonction chercher(x, T)
    n ≔ longueur(T)
    i ≔ 0
    j ≔ n − 1
tant que i < j faire
    m ≔ (i + j) ÷ 2
    si T[m] = x alors
        retourner m
    sinon si x < T[m] alors
        i ≔ m + 1
    sinon
        j ≔ m − 1
retourner −1
```

Terminaison ?

Correction ?

Efficacité ?