

Introduction aux algorithmes III

Algorithmes et structures de données, 2025-2026

P. Albuquerque (B410) et O. Malaspinas (A401), ISC, HEPIA
2025-09-30

En partie inspiré des supports de cours de P. Albuquerque

Rappel (1/2)

Quels algos avons-nous vu la semaine passée ?

Rappel (1/2)

Quels algos avons-nous vu la semaine passée ?

- L'algorithme de la factorielle.
- L'algorithme du PPCM.

Algorithme du PPCM ?

Rappel (2/2)

Algorithme du PPCM ?

```
int main() {  
    int m = 15, n = 12;  
    int mult_m = m, mult_n = n;  
    while (mult_m != mult_n) {  
        if (mult_m > mult_n) {  
            mult_n += n;  
        } else {  
            mult_m += m;  
        }  
    }  
    printf("Le ppcm de %d et %d est %d\n", n, m, mult_m);  
}
```

Le calcul du PPCM (1/2)

Réusinage : Comment décrire une fonction qui ferait ce calcul (arguments, sorties) ?

Le calcul du PPCM (1/2)

Réusinage : Comment décrire une fonction qui ferait ce calcul (arguments, sorties) ?

En C on pourrait la décrire comme

```
int ppcm(int a, int b); // La **signature** de cette fonction
```

Le calcul du PPCM (1/2)

Réusinage : Comment décrire une fonction qui ferait ce calcul (arguments, sorties) ?

En C on pourrait la décrire comme

```
int ppcm(int a, int b); // La **signature** de cette fonction
```

Algorithme

Par groupe de 3 (5-10min) :

- réfléchissez à un algorithme alternatif donnant le PPCM de deux nombres ;
- écrivez l'algorithme en pseudo-code.

Le calcul du PPCM (1/2)

Réusinage : Comment décrire une fonction qui ferait ce calcul (arguments, sorties) ?

En C on pourrait la décrire comme

```
int ppcm(int a, int b); // La **signature** de cette fonction
```

Algorithme

Par groupe de 3 (5-10min) :

- réfléchissez à un algorithme alternatif donnant le PPCM de deux nombres ;
- écrivez l'algorithme en pseudo-code.
- Si un nombre, p , est multiple de a et de b alors il peut s'écrire $p = a * i = b * j$ ou encore $p / a = i$ et $p / b = j$.

Le calcul du PPCM (2/2)

Pseudo-code

```
int ppcm(int a, int b) {  
    for (i in [1, b]) {  
        if a * i est divisible par b {  
            return a * i  
        }  
    }  
}
```

Le code du PPCM de 2 nombres (1/2)

Implémentez le pseudo-code et postez le code sur matrix (5min).

Le code du PPCM de 2 nombres (1/2)

Implémentez le pseudo-code et postez le code sur matrix (5min).

Un corrigé possible

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int n = 15, m = 12;
    int i = 1;
    while (n * i % m != 0) {
        i++;
    }
    printf("Le ppcm de %d et %d est %d\n", n, m, n*i);
}
```

Le code du PPCM de 2 nombres (2/2)

Corrigé alternatif

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int res = n*m;
    for (int i = 2; i <= m; i++) {
        if (n * i % m == 0) {
            res = n * i;
            break;
        }
    }
    printf("Le ppcm de %d et %d est %d\n", n, m, res);
}
```

Le calcul du PGCD (1/5)

Définition

Le plus grand commun diviseur (PGCD) de deux nombres entiers non nuls est le plus grand entier qui les divise en même temps.

Exemples :

$$\text{PGCD}(3, 4) = 1,$$

$$\text{PGCD}(4, 6) = 2,$$

$$\text{PGCD}(5, 15) = 5.$$

Le calcul du PGCD (1/5)

Définition

Le plus grand commun diviseur (PGCD) de deux nombres entiers non nuls est le plus grand entier qui les divise en même temps.

Exemples :

$$\text{PGCD}(3, 4) = 1,$$

$$\text{PGCD}(4, 6) = 2,$$

$$\text{PGCD}(5, 15) = 5.$$

Mathématiquement

Décomposition en nombres premiers :

$$36 = 2^2 \cdot 3^2, \quad 90 = 2 \cdot 5 \cdot 3^2,$$

On garde tous les premiers à la puissance la plus basse

$$\text{PGCD}(36, 90) = 2^{\min 1, 2} \cdot 3^{\min 2, 2} \cdot 5^{\min 0, 1} = 18.$$

Le calcul du PGCD (2/5)

Algorithme

Par groupe de 3 (5-10min) :

- réfléchissez à un algorithme alternatif donnant le PGCD de deux nombres ;
- écrivez l'algorithme en pseudo-code.

Le calcul du PGCD (2/5)

Algorithme

Par groupe de 3 (5-10min) :

- réfléchissez à un algorithme alternatif donnant le PGCD de deux nombres ;
- écrivez l'algorithme en pseudo-code.

Exemple d'algorithme

PGCD(36, 90) :

```
90 % 36 != 0 // otherwise 36 would be PGCD
90 % 35 != 0 // otherwise 35 would be PGCD
90 % 34 != 0 // otherwise 34 would be PGCD
...
90 % 19 != 0 // otherwise 19 would be PGCD
90 % 18 == 0 // The end!
```

- 18 modulus, 18 assignments, et 18 comparaisons.

Le calcul du PGCD (3/5)

Transcrivez cet exemple en algorithme et codez-le (5-10min) !

Le calcul du PGCD (3/5)

Transcrivez cet exemple en algorithme et codez-le (5-10min) !

Optimisation

- Combien d'additions / comparaisons au pire ?
- Un moyen de le rendre plus efficace ?

Le calcul du PGCD (3/5)

Transcrivez cet exemple en algorithme et codez-le (5-10min) !

Optimisation

- Combien d'additions / comparaisons au pire ?
- Un moyen de le rendre plus efficace ?

Tentative de correction

```
void main() {  
    int n = 90, m = 78;  
    int gcd = 1;  
    for (int div = n; div >= 2; div--) { // div = m, sqrt(n)  
        if (n % div == 0 && m % div == 0) {  
            gcd = div;  
            break;  
        }  
    }  
    printf("Le pgcd de %d et %d est %d\n", n, m, gcd);  
}
```

Le calcul du PGCD (4/5)

Réusinage : l'algorithme d'Euclide

Dividende = Diviseur * Quotient + Reste

PGCD(35, 60):

$35 = 60 * 0 + 35$ // 60 \rightarrow 35, 35 \rightarrow 60

$60 = 35 * 1 + 25$ // 35 \rightarrow 60, 25 \rightarrow 35

$35 = 25 * 1 + 10$ // 25 \rightarrow 35, 20 \rightarrow 25

$25 = 10 * 2 + 5$ // 10 \rightarrow 25, 5 \rightarrow 10

$10 = 5 * 2 + 0$ // PGCD = 5!

Le calcul du PGCD (4/5)

Réusinage : l'algorithme d'Euclide

Dividende = Diviseur * Quotient + Reste

PGCD(35, 60):

35 = 60 * 0 + 35 // 60 -> 35, 35 -> 60

60 = 35 * 1 + 25 // 35 -> 60, 25 -> 35

35 = 25 * 1 + 10 // 25 -> 35, 20 -> 25

25 = 10 * 2 + 5 // 10 -> 25, 5 -> 10

10 = 5 * 2 + 0 // PGCD = 5!

Algorithme

Par groupe de 3 (5-10min) :

- analysez l'exemple ci-dessus ;
- transcrivez le en pseudo-code.

Le calcul du PGCD (5/5)

Pseudo-code

```
entier pgcd(m, n)
    tmp_n = n
    tmp_m = m
    tant que (tmp_m ne divise pas tmp_n)
        tmp = tmp_n
        tmp_n = tmp_m
        tmp_m = tmp modulo tmp_m
    retourne tmp_m
```

Le code du PGCD de 2 nombres

Implémentez le pseudo-code et postez le code sur matrix (5min).

Le code du PGCD de 2 nombres

Implémentez le pseudo-code et postez le code sur matrix (5min).

Un corrigé possible

```
#include <stdio.h>
void main() {
    int n = 90;
    int m = 78;
    printf("n = %d et m = %d\n", n, m);
    int tmp_n = n;
    int tmp_m = m;
    while (tmp_n%tmp_m > 0) {
        int tmp = tmp_n;
        tmp_n = tmp_m;
        tmp_m = tmp % tmp_m;
    }
    printf("Le pgcd de %d et %d est %d\n", n, m, tmp_m);
}
```

Quelques algorithmes simples

- Stockage d'une liste de nombre et recherche de la valeur minimale
- Anagrammes
- Palindromes
- Crible d'Ératosthène

Quelques algorithmes simples

- Stockage d'une liste de nombre et recherche de la valeur minimale
- Anagrammes
- Palindromes
- Crible d'Ératosthène
- Ces algorithmes nécessitent d'utiliser des **tableaux**.

Collections : tableaux statiques

- Objets de même type : leur nombre est **connu à la compilation**
- Stockés de façon contiguë en mémoire (très efficace)

```
#define SIZE 10  
int entiers[] = {2, 1, 4, 5, 7}; // taille 5, initialisé  
int tab[3]; // taille 3, non initialisé  
float many_floats[SIZE]; // taille 10, non initialisé
```

- Les indices sont numérotés de 0 à taille-1;

```
int premier = entier[0]; // premier = 2  
int dernier = entier[4]; // dernier = 7
```

- Les tableaux sont **non-initialisés** par défaut;
- Les bornes ne sont **jamais** vérifiées.

```
int indetermine_1 = tab[1]; // undefined behavior  
int indetermine_2 = entiers[5]; // UB
```

- Depuis C99 la taille peut être *inconnue à la compilation* (VLA);

```
int size;  
scanf("%d", &size);  
char string[size];
```

Remarques

- Depuis C99 la taille peut être *inconnue à la compilation* (VLA);

```
int size;  
scanf("%d", &size);  
char string[size];
```

- Considéré comme une mauvaise pratique : que se passe-t-il si `size == 1e9` ?
- On préfère utiliser l'allocation **dynamique** de mémoire pour ce genre de cas-là (spoiler du futur du cours).

Initialisation

- Les variables ne sont **jamais** initialisées en C par défaut.
- Question : Que contient le tableau suivant ?

```
double tab[4];
```

Initialisation

- Les variables ne sont **jamais** initialisées en C par défaut.
- Question : Que contient le tableau suivant ?

```
double tab[4];
```

- Réponse : On en sait absolument rien !
- Comment initialiser un tableau ?

Initialisation

- Les variables ne sont **jamais** initialisées en C par défaut.
- Question : Que contient le tableau suivant ?

```
double tab[4];
```

- Réponse : On en sait absolument rien !
- Comment initialiser un tableau ?

```
#define SIZE 10
double tab[SIZE];
for (int i = 0; i < SIZE; ++i) {
    tab[i] = rand() / (double)RAND_MAX * 10.0 - 5.0;
    // tab[i] contient un double dans [-5;5]
}
int other_tab[4] = {0}; // pareil que {0, 0, 0, 0}
```

Recherche du minimum dans un tableau (1/2)

Problématique

Trouver la valeur minimale contenue dans un tableau et l'indice de l'élément le plus petit.

Écrire un pseudo-code résolvant ce problème (groupe de 3), 2min

Recherche du minimum dans un tableau (1/2)

Problématique

Trouver la valeur minimale contenue dans un tableau et l'indice de l'élément le plus petit.

Écrire un pseudo-code résolvant ce problème (groupe de 3), 2min

```
index = 0
min   = tab[0]
pour i de 1 à SIZE - 1
    si min > tab[i]
        min = tab[i]
        index = i
```

Recherche du minimum dans un tableau (2/2)

Implémenter ce bout de code en C (groupe de 3), 4min

Recherche du minimum dans un tableau (2/2)

Implémenter ce bout de code en C (groupe de 3), 4min

```
int index = 0;
float min = tab[0];
for (int i = 1; i < SIZE; ++i) {
    if (min > tab[i]) {
        min = tab[i];
        index = i;
    }
}
```