

Complexité expérimentale

- Calcul « théorique » compliqué (?)
- Cas moyen, pire cas, et en pratique
 - Exemple « deep learning » : en théorie c'est mauvais, en pratique ça marche très bien
(reconnaissance de visages, parole,).

Comment mesurer une complexité en pratique (exemple un tri) ?

Principe

- Point de départ:
 - On a un algorithme (programmé) dont on veut estimer la complexité
- On ajoute des compteurs de tests, d'instructions
- On recueille les valeurs des compteurs en fonction de données dont la taille N varie
- Ensuite estimation (régression,)

Exemple concret (polynome)

- Soit le code:

```
public double horner(double x) {  
    double resultat = coef[deg];  
    for (int i = deg - 1; i >= 0; i--) {  
  
        resultat = resultat * x + coef[i];  
    }  
    return resultat;  
}
```

On veut connaître le nombre d'opérations (pour comparer avec une autre méthode de calcul)

Se définir un (des) compteur(s)

*On se définit autant de compteurs
(attributs et méthodes) que de
grandeurs à mesurer*

```
private int nbOp;  
private void init(){  
    nbOp=0;  
}
```

On instrumente le code

```
public double horner(double x) {  
    double resultat = coef[deg];  
    for (int i = deg - 1; i >= 0; i--) {  
        resultat = resultat * x + coef[i];  
    }  
    return resultat;  
}
```

```
public double horner(double x) {  
    double resultat = coef[deg];  
    for (int i = deg - 1; i >= 0; i--) {  
        > nbOp++;  
        resultat = resultat * x + coef[i];  
    }  
    return resultat;  
}
```

Ensuite on utilise

```
public void testHorner() {  
    init();  
    double v = horner(2.5);  
    System.out.println(nbOp);  
}
```

Un 2^{ème} exemple

- Cas de fonction récursive (mais le principe reste inchangé)

Exemple concret

- Soit le code

```
public static int fibo(int n) {  
    if (n == 0) {  
        return 1;  
    }  
    if (n == 1) {  
        return 1;  
    }  
    return fibo(n - 1) + fibo(n - 2);  
}
```


Compteurs et outillages

```
public static int compteur;  
public static void inc(){  
    compteur++;  
}  
public static void inc(int nb){  
    compteur += nb;  
}  
public static void init(){  
    compteur = 0;  
}  
public static int recap(){  
    return compteur;  
}
```

*On se définit autant de compteurs
(attributs et méthodes) que de
grandeurs à mesurer*

*Dans quelle classe se trouve
ces méthodes et attributs ?*

Instrumentations du code

```
public static int fibo(int n) {  
    if (n == 0) {  
        return 1;  
    }  
    if (n == 1) {  
        return 1;  
    }  
    return fibo(n - 1) + fibo(n - 2);  
}  
  
public static int fibo(int n) {  
    if (n == 0) {  
        inc();  
        return 1;  
    }  
    if (n == 1) {  
        inc();  
        return 1;  
    }  
    inc(3); // 2 appels et une addition  
    return fibo(n - 1) + fibo(n - 2);  
}
```

The diagram illustrates the instrumentation of a Fibonacci function. It shows two versions of the function side-by-side. The left version is the original code, and the right version is the instrumented code. Blue arrows indicate the insertion of instrumentation code (in blue) at specific points in the original code (in purple). The instrumentation includes calls to `inc()` at the start of the base cases and before the recursive call, and a call to `inc(3)` with a comment `// 2 appels et une addition` before the final return statement.

Usage

```
for (int i = 0; i < 100; i++) {  
    Exemple2.init();  
    Exemple2.fibo(i);  
    System.out.println( Exemple2.recap());  
}
```

Code dans la classe Exemple2 !

Ensuite il faut traiter les résultats

