Algorithmique et structures de données

Loïc Demange
loic.demange@etud.univ-paris8.fr

25 septembre 2020

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

TP précédent

Objectifs du TP :

- Écrire du code, compiler, et exécuter le programme.
- Créer une fonction addition qui renvoie l'addition de deux entiers, et faire en sorte d'afficher le résultat.
- Faire en sorte de réaliser cette addition tant que l'entier est inférieur à 100.
- Faire en sorte de réaliser cette addition que 10 fois, et si l'entier est inférieur à 50 rajouter 75.
- [BONUS] Implémenter la suite de Fibonacci.

TP précédent

Créer une fonction addition qui renvoie l'addition de deux entiers, et faire en sorte d'afficher le résultat.

```
#include <stdio.h>
int add(int a, int b)
{
    return a + b;
}

int main()
{
    int a = 10, b = 20;
    printf("%d\n", add(a,b));
    return 0;
}
```

TP précédent

Faire en sorte de réaliser cette addition tant que l'entier est inférieur à 100.

```
int main()
{
    int a = 10, b = 20, res = 0;
    while(res < 100)
        res += add(a,b);
    printf("%d\n", res);
    return 0;
}
```

TP précédent

Faire en sorte de réaliser cette addition que 10 fois, et si l'entier est inférieur à 50 rajouter 75.

```
int main()
{
    int a = 10, b = 20, res = 0;
    for(int i = 0; i < 10; i++)
        res += add(a,b);
    if(res < 50)
        res += 75;
    printf("%d\n", res);
    return 0;
}
```

TP précédent

[BONUS] Implémenter la suite de Fibonacci.

```
int fibo(int n)
{
    int f0 = 0, f1 = 1, t = 0;
    for(int i = 0; i < n; i++)
    {
        t = f0 + f1;
        f0 = f1;
        f1 = t;
    }
    return f0;
}
```

Complexité

- Comment mesurer les performances d'un algorithme ?

Complexité

- Comment mesurer les performances d'un algorithme ?

Mesurer le temps est une mauvaise idée puisque cela dépend de la cible. Calculer le nombre d'instructions élémentaires en fonction de l'entrée est plus efficace.

Remarque Le nombre d'instructions ne donne qu'un ordre d'idée relativement fiable, le comportement de la cible ayant quand même un impact.

Complexité

- Qu'est-ce qu'une instruction élémentaire ?

Complexité

- Qu'est-ce qu'une instruction élémentaire ?

Cela dépend de notre objectif et de notre précision.

Pour le moment, nous allons définir comme étant instruction élémentaire les instructions dites "basiques" (déclaration, affectation, retour, addition, soustraction, comparaison, etc.).

Complexité

Combien y a-t-il d'instructions élémentaires ?

```
int main()
{
    int x = a + b;
    return 0;
}
```

Complexité

Combien y a-t-il d'instructions élémentaires ?

```
int main()
{
    int x = 2 + 3;
    return 0;
}
```

Avec notre définition d'instructions élémentaires, il y a 4 instructions : déclaration, affectation, addition et retour.

Remarque Il s'agit ici d'instructions purement théoriques. En réalité, le compilateur va réaliser l'addition et l'affectation à la compilation.

Combien y a-t-il d'instructions élémentaires ?

```
int main()
{
    int n = 4;
    int x = 0;
    for(int i = 0; i < n; i++)
    {
        x += 2;
    }
    return 0;
}
```

Combien y a-t-il d'instructions élémentaires ?

```
int main()
{
    int n = 4;
    int x = 0;
    for(int i = 0; i < n; i++)
    {
        x += 2;
    }
    return 0;
}
```

Il y a ici 3 déclarations et affectations, n comparaisons, $n \cdot (1 + 1)$ additions et affectations et un retour, soit 27 instructions.

Combien y a-t-il d'instructions élémentaires ?

```
int fibo(int n)
{
    int f0 = 0, f1 = 1, t = 0;
    for(int i = 0; i < n; i++)
    {
        t = f0 + f1;
        f0 = f1;
        f1 = t;
    }
    return f0;
}
```

Complexité

Combien y a-t-il d'instructions élémentaires ?

```
int fibo(int n)
{
    int f0 = 0, f1 = 1, t = 0;
    for(int i = 0; i < n; i++)
    {
        t = f0 + f1;
        f0 = f1;
        f1 = t;
    }
    return f0;
}
```

Il y a ici 4 déclarations et affectations, n comparaisons, $n \cdot (1+1)$ additions et affectations, $n \cdot (1+1)$ affectations et un retour, soit $7n+9$ instructions.
Remarque On ne connait pas n à l'avance, la complexité sera donc sous la forme d'un polynôme en fonction de l'entrée.

Complexité

Grand-O

Soit f une fonction.
On définit $\mathcal{O}(f(x))$ l'ensemble des fonctions $g(x)$ telles qu'il existe une constante positive réelle c et un entier non négatif x_0 tels que pour tout $x > x_0$, $g(x) < c \cdot f(x)$.

Question Donnez un grand-O qui englobe la complexité de la suite de Fibonacci $(7n + 4)$.

Complexité

Grand-O

Soit f une fonction.
On définit $\mathcal{O}(f(x))$ l'ensemble des fonctions $g(x)$ telles qu'il existe une constante positive réelle c et un entier non négatif x_0 tels que pour tout $x > x_0$, $g(x) < c \cdot f(x)$.

Question Donnez un grand-O qui englobe la complexité de la suite de Fibonacci $(7n + 4)$. $\mathcal{O}(n)$.

Complexité

De manière plus simple, grand-O va isoler la forme générale de la fonction. Par exemple, si $f(n) = 30n^3 + 2n^2 + 1000$, alors $f(n) \in \mathcal{O}(n^3)$.

On appelle ça la complexité asymptotique, car il s'agit d'une approximation qui s'affine lorsque l'entrée (ici n) tend vers l'infini.

TP

Objectifs du TP :

- Coder le jeu “Plus ou moins”, où l'ordinateur doit trouver le nombre auquel vous pensez.
- Exprimer sa complexité asymptotique à l'aide du grand-O.

Indication : ce code va stocker une entrée utilisateur (ici un caractère) dans la variable *c*.

```
#include <stdio.h>
int main()
{
    char c;
    int ret = scanf("%c", &c);
    if(ret == 1)...
    return 0;
}
```

Seules trois entrées sont nécessaires : ‘+’, ‘-’, ‘=’.