TD 2 : Récursivité

3 décembre 2023

Exercice 1

```
int somme(int n){
    if (n==0){
        return n;
    }
    else{
        return n+somme(n-1);
    }
}
```

Exercice 2

1)

```
float suite(float u0, int n){
    if (n==0){
        return u0;
    }
    else{
        return 0.5*suite(u0,n-1)+3;
    }
}
```

2)

```
float suite_fonction(float u0, int n){
    if (n==0){
        return u0;
    }
    else{
        return f(suite(u0,n-1));
    }
}
```

Exercice 3

Rappel : soient $0 \leq p \leq n$:

— Si $p = 0$ ou $p = n$, alors $\binom{n}{p} = 1$

— Sinon, $\binom{n}{p} = \binom{n-1}{p} + \binom{n-1}{p-1}$

Ecrire une fonction qui prend en entrée p et n avec $0 \leq p \leq n$ et calcule récursivement le coefficient binomial $\binom{n}{p}$.

```
int coeff_binom(int p, int n){
    if (p==0 || p==n){
        return 1;
    }
    else{
        return coeff_binom(p-1,n-1) + coeff_binom(p,n-1);
    }
}
```

Exercice 4

1) Soient a, b deux entiers naturels avec $b \neq 0$.

— $PGCD(a, b) = b$ si le reste de la division euclidienne de a par b vaut 0

— $PGCD(a, b) = PGCD(b, a \mod b)$ sinon

2)

```
int pgcd(int a, int b){
    if (a % b == 0){
        return b;
    }
    else{
        return pgcd(b, a%b);
    }
}
```

Exercice 5

On note u_n le nombre d'appels récursifs de `fibo(n)` :

— Si $n = 0$ ou $n = 1$, $u_n = 1$

— Si $n \geq 2$, on a l'appel initiale (1) et l'ensemble des appels récursifs engendrés par `fibo(n-1)` et `fibo(n-2)` : on a donc la relation $u_n = u_{n-1} + u_{n-2} + 1$.

On va chercher à construire à partied de u_n une suite v_n de la forme $u_n + k$ avec k constant, vérifiant la relation $v_n = v_{n-1} + v_{n-2}$ pour $n \geq 2$, comme pour la suite de Fibonacci.

Analyse : soit v_n de la forme $u_n + k$ avec k constant, vérifiant la relation $v_n = v_{n-1} + v_{n-2}$ pour $n > 2$. On a :

$v_n = u_n + k = u_{n-1} + u_{n-2} + 1 + k = v_{n-1} + v_{n-2} - 2k + 1 + k = v_{n-1} + v_{n-2} - k + 1$

On a donc : $v_n = v_{n-1} + v_{n-2} \Rightarrow -k + 1 = 0 \Rightarrow k = 1$.

Synthèse : soit $v_n = u_n + 1$, alors pour $n \geq 2$, on a $v_n = u_n + 1 = u_{n-1} + u_{n-2} + 1 + 1 = v_{n-1} + v_{n-2}$. La suite v_n vérifie la relation recherchée.

Soit F_n la suite de Fibonacci initialisée par $F_0 = 1$ et $F_1 = 1$. Montrons que $v_n = 2F_n$.

Initialisation : on a $v_0 = u_0 + 1 = 2 = 2F_0$ et $v_1 = u_1 + 1 = 2 = 2F_0$.

Récurrence : soit $n \geq 0$ tel que $v_n = 2F_n$ et $v_{n+1} = 2F_{n+1}$. Alors $v_{n+2} = v_{n+1} + v_n = 2F_{n+1} + 2F_n = 2F_{n+2}$.

On a donc $u_n + 1 = v_n = 2F_n$, d'où $u_n = 2F_n - 1$ pour tout $n \geq 0$.

Exercice 6

1) On utilise les notations suivantes :

— $L[i]$ pour le i-ème terme de la liste L (en indexant à partir de 0)

— $L[i, j]$ pour la sous-liste de L constituée des termes entre le i-ème et le $j - 1$-ème

— $+$ pour la concaténation de deux listes

Algorithme 1 Fusion

Entrée: Liste L_1, liste L_2

Sortie: Fusion triée de L_1 et L_2

```
si L1 est vide alors
    retourne L2
sinon si L2 est vide alors
    retourne L1
sinon

    si L1[0] < L2[0] alors
        retourne [L1[0]] + Fusion(L1[1, :], L2)

    sinon
        retourne [L2[0]] + Fusion(L1, L2[1, :])

    fin si
fin si
```

2)

Algorithme 2 Tri Fusion

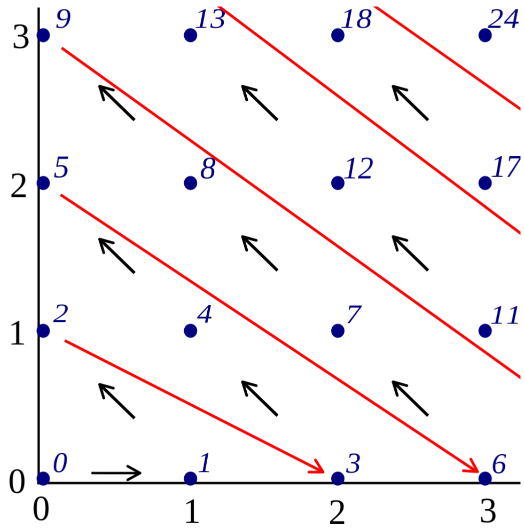
Entrée: Liste L

Sortie: Liste L triée

```
si L est vide ou L est de longueur 1 alors
    retourne L
sinon
    retourne Fusion(Tri Fusion(L[:, ⌊Longueur(L)/2⌋]), Tri Fusion(L[⌊Longueur(L)/2⌋, :]))
fin si
```

Exercice 7

1) Le schéma suivant (source : Wikipédia) présente l'ordre dans lequel sont parcourues les différents couples d'entiers de $\mathbb{N}^2$.



2) Appelons ϕ la bijection entre $\mathbb{N}^2$ et $\mathbb{N}$. La façon récursive de définir cette bijection est la suivante, pour un couple d'entier (n_1, n_2) :

- Cas terminal : si $n_1 = n_2 = 0$, alors $\phi(n_1, n_2) = 0$
- Sinon, $\phi(n_1, n_2) = 1 + \phi(n_1', n_2')$ où (n_1', n_2') est le couple précédent dans le parcours de diagonale :
  - Si $n_2 = 0$, on est au début d'une nouvelle diagonale parcourue : le couple d'entiers qui précède dans le parcours de diagonales est la fin de la diagonale précédente, c'est-à-dire le couple $(0, n_1 - 1)$
  - Sinon, le prédécesseur se trouve sur la même diagonale, en bas à droite : c'est le couple $(n_1 + 1, n_2 - 1)$.

L'implémentation en C de la bijection est la suivante :

```c
int bijection(int n1, int n2){
    if (n1 == 0 && n2 == 0){
        return 0;
    }
    else{
        if (n2 == 0){
            return 1 + bijection(0,n1-1);
        }
        else{
            return 1+bijection(n1+1,n2-1);
        }
    }
}
```