Calcul des nombres de Fibonacci [cx03] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 21 mai 2018

Table des matières

Python - Calcul des nombres de Fibonacci (Solution)

Mots-Clés Complexité des algorithmes ■
Requis Axiomatique impérative, Récursivité des actions ■
Difficulté • ○ ○

Objectif
Cet exercice analyse la complexité de la suite de Fibonacci.

1 Calcul des nombres de Fibonacci / pgfib

Définition
Les **nombres de Fibonacci** sont définis par la relation de récurrence :

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2}, \quad n \geq 2 \end{cases}$$

Nombre d'or

On peut montrer que F_n est l'entier le plus proche de $\Phi^n/\sqrt{5}$ avec Φ le nombre d'or ($\Phi = (1+\sqrt{5})/2$).

1.1 Algorithme itératif

Écrivez une fonction `fibIter(n)` qui calcule et renvoie le `n`-eme nombre de FIBONACCI en utilisant la récurrence.

Validez votre fonction avec la solution.

Solution Python @[pgfib.py]

```
def fibIter(n):
   f0 = 0
   f1 = 1
   fn = -1
   if (n == 0):
      fn = f0
   if (n == 1):
      fn = f1
   for k in range(2,n+1):
      fn = f1 + f0
      f0,f1 = f1,fn
   return fn
```

Quelle est la complexité de l'algorithme ?

Solution simple
Clairement, l'algorithme itératif est en $\Theta(n)$.

1.2 Algorithme récursif naïf

Écrivez une fonction récursive `fibRec(n)` qui calcule et renvoie le `n`-eme nombre de FIBONACCI en utilisant la récurrence.

Validez votre fonction avec la solution.

Solution Python @[pgfib.py]

```
def fibRec(n):
   if (n == 0):
      return 0
   elif (n == 1):
      return 1
   else:
      return fibRec(n-1) + fibRec(n-2)
```

Montrez par récurrence que la complexité (en nombre d'additions) de cet algorithme est en $\Omega(2^{n/2})$.

Solution simple

On veut montrer qu'il existe une constante c strictement positive telle que $T(n) \geq c2^{n/2}$, pour des valeurs de n supérieures à une certaine borne n_0 (à déterminer).

Supposons le résultat démontré jusqu'au rang $n-1$. Alors :

$$\begin{aligned} T(n) &= T(n-1) + T(n-2) + 1 \\ &\geq c2^{(n-1)/2} + c2^{(n-2)/2} + 1 \\ &\geq c2^{(n-2)/2} + c2^{(n-2)/2} + 1 \\ &\geq 2 \times c2^{(n-2)/2} \\ &= c2^{n/2} \end{aligned}$$

Il nous reste à montrer que cette équation est vraie « au départ ». Nous ne pouvons bien évidemment pas partir des cas $n = 0$ et $n = 1$, puisque pour ces valeurs $T(n) = 0$. Nous partons donc des cas $n = 2$ et $n = 3$ (la récurrence nécessite **deux** valeurs de départ) :

- Cas $n = 2$: `fibRec(2)=fibRec(1)+fibRec(0)` et $T(2) = 1$. Pour que la propriété désirée soit vraie, c doit vérifier :

$$1 \geq c2^{2/2} = 2c \quad \Leftrightarrow \quad c \leq \frac{1}{2}$$

- Cas $n = 3$: `fibRec(3)=fibRec(2)+fibRec(1)` et $T(3) = 2$. Pour que la propriété désirée soit vraie, c doit vérifier :

$$2 \geq c2^{3/2} = 2\sqrt{2}c \quad \Leftrightarrow c \leq \frac{\sqrt{2}}{2}$$

Donc si $c = \frac{1}{2}$, pour $n \geq 2$, on a $T(n) \geq c2^{n/2}$ et donc $T(n) = \Omega(2^{n/2})$.

Remarque

On peut montrer (par récurrence) que le nombre d'appels pour calculer F_n est F_{n+1}. Comme, en dehors des appels récursifs, l'exécution du corps de la fonction est en $\Theta(1)$, la complexité est donc en $\Theta(\Phi^n)$.

1.3 Algorithme récursif terminal

Écrivez une fonction récursive `fib(n)` qui calcule et renvoie le `n`-eme nombre de FIBONACCI en utilisant la récursivité terminale.

Validez votre fonction avec la solution.

Solution Python @[pgfib.py]

```
def fibRt(n,a,b):
   return a if (n == 1) else fibRt(n-1, a+b, a)
```

```
def fib(n):
   return 0 if (n == 0) else fibRt(n, 1, 0)
```

Quelle est la complexité (en nombre d'additions) de cet algorithme ?

Solution simple
La complexité de l'algorithme `fibRt`, en nombre d'additions, est donnée par la récurrence $T(n) = 1 + T(n-1)$. On a donc $T(n) = n - 1$ pour `fibRt`, et par extension pour la nouvelle version de `fib`.

1.4 Conclusion

Des divers algorithmes, que pouvez dire ?

Solution simple
L'algorithme naïf récursif est impraticable tandis que l'algorithme itératif et l'algorithme récursif terminal sont efficaces.

2 Références générales

Comprend ■