exercice_xn

February 24, 2021

1 1A.algo - Calculer x**n le plus rapidement possible

C'est un exercice courant lors des entretiens d'embauche. Il faut savoir ce qu'est la dichotomie et la notation binaire d'un nombre.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.0.1 Enoncé

Comme n est entier, la façon la plus simple est de calculer $x \times x \times ... \times x$ mais existe-t-il plus rapide que cela ?

1.0.2 Solution

L'idée de départ consiste à écrire $x^{2n} = (x^n)^2$. En extrapolant, on en déduit que si $n = 2^k$, alors le coût du calcul de x^n consistera en k itérations en on 2^k.

[2]:
```
def puissance2k(x,k):
    while k > 0 :
        x *= x
        k -= 1
    return x

for i in range(0,4) :
    print ( "2^(2^{0})=2^{1}={2}".format( i, 2**i, puissance2k ( 2, i ) ) )
```

```
2^(2^0)=2^1=2
2^(2^1)=2^2=4
2^(2^2)=2^4=16
2^(2^3)=2^8=256
```

Lorsque n n'est pas une puissance de 2, il suffit que le décomposer en écriture binaire. Si $n = \sum_k a_k 2^k$, avec $a_k \in \{0, 1\}$, alors $x^n = \prod_k x^{a_k 2^k}$.

[3]:
```
def puissance(x,n):
    r = 1
    while n > 0 :
        if n % 2 == 1 : r *= x
        x *= x
        n //= 2
```

```
    return r

for i in range(0,9) :
    print("2^{0}={1}".format(i, puissance( 2, i)))
```

```
2^0=1
2^1=2
2^2=4
2^3=8
2^4=16
2^5=32
2^6=64
2^7=128
2^8=256
```

[4]: