correction_TP2

October 13, 2018

Activité 1

Q1

Si l'écriture d'un entier p en base 2 est $p = a_0 a_1 \dots a_k$, alors on obtient son expression décimale en écrivant :

$p = \sum_{k=0}^{n} a_{n-k} 2^k$. Nous devons donc écrire un algorithme

-qui demande à l'utilisateur de saisir une chaine de caractère constituée de 0 et de 1 (écriture binaire de p)

-qui récupère le nombre de caractères (valeur de n)

-qui calcule la somme demandée à l'aide d'une boucle for en ajoutant à chaque fois la valeur obtenue précédemment, notée some, la nouvelle valeur ($a_{n_k} 2^k$) que l'on stocke encore dans la même variable.

-qui affiche la valeur de cette somme.

Q3

```
In [2]: nbre2=input("tape un nombre ecrit en base 2 : ")
        n=len(nbre2)-1
        somme=0
        for k in range(0,n+1):
            ak=int(nbre2[n-k])
            somme=somme+ak*2**(k)


        print("{:1} en base 10 est égal à {:2}".format(nbre2,somme))
```

```
tape un nombre ecrit en base 2 : 1111
1111 en base 10 est égal à 15
```

Activité 2

Q1

On fait des divisions euclidiennes successives par 2 du nombre de départ p exprimé en base 10 jusque'à obtenir un quotient nul. On obtient une collection de restes $r_0, r_1, \dots, r_n$ de restes. On sait qu'alors l'écriture en binaire de p est $r_n \dots r_0$.

Q2

```
In [3]: nbre10=int(input("entre un nombre ecrit en base 10 :  "))
        nbre2=""       #nbre2 doit sortir à la fin de l'algorithme la valeur binaire
        dividende=nbre10    #Le dividende de la première division euclidienne est l
```

```
        while dividende>0:
            r=dividende%2    #r est le reste de la division euclidienne efectuée à
            dividende=dividende//2 # A chaque étape, on fait la division euclidienn
            nbre2=str(r)+nbre2  #On met le reste obtenu en tête de la chaîne de ca

        print(nbre2)

entre un nombre ecrit en base 10 :  8
1000
```

Q3

```
In [4]: bin(8)

Out[4]: '0b1000'
```

On "valide" sur un exemple l'algorithme proposé, ce qui ne prouve bien évidemment sa vraie exactitude mais prouve au moins qu'il fonctionne sur certains exemples (et plus il fonctionne sur des exemples, plus nous sommes en mesure d'estimer qu'il fonctionne).

Activité 3

Q1

On commence par estimer la plus grande puissance de deux du nombre p exprimé en base 10 et on calcule leur différence. On itère ce principe en testant à chaque fois la différence obtenue avec la puissance de deux précédente. Si la différence renvoie un nombre négatif strictement négatif, on pose : $a_k = 0$, sinon on pose $a_k = 1$. Alors, l'écriture binaire de p est : $a_0 \dots a_n$.

Q2

```
In [5]: #fonction plus grande puissance de 2
        def puiss2(n):
            k=0
            while 2**k <=n:
                k=k+1
            return(2**(k-1))

In [6]: puiss2(15)

Out[6]: 8
```

Q3

```
In [8]: def base10to2puiss(n):
            nbre=n
            k=puiss2(n)
            nbre2=""
            while k >=1:
                if k>nbre :
                    nbre2=nbre2+"0"
                    nbre=nbre
```

```
            k=k/2
        else :
            nbre2=nbre2+"1"
            nbre=nbre-k
            k=k/2
    return(nbre2)
```

```
In [9]: base10to2puiss(15)
```

```
Out[9]: '1111'
```

```
In [10]: bin(15)
```

```
Out[10]: '0b1111'
```

Activité 1 bis

Si $a_0 \dots a_n$ est l'écriture d'un entier p, alors nous avons : $p = (\dots(((2 \times a_0) + a_1) \times 2 + a_2) \times 2 + \dots) \times 2 + a_n$.

Par exemple : $15 = ((1 \times 2 + 1) \times 2 + 1) \times 2 + 1$ et 15 s'écrit 1111 en binaire.

Au final, à chaque étape, on fait le même calcul, à savoir multiplier le résultat au rang k par 2 et ajouter la valeur de a_{k+1} afin d'obtenir le résultat au rang $k + 1$. Il s'agit de la méthode de Horner.

```
In [11]: nbre2=input("entre un nombre écrit en base 2 :  ")
         n=len(nbre2)
         facteur=int(nbre2[0])
         for k  in range(0,n-1) :
             facteur=2*facteur+int(nbre2[k+1])

         print("{:1} en base 10 est égal à {:2}".format(nbre2,facteur))
```

```
entre un nombre écrit en base 2 :  1111
1111 en base 10 est égal à 15
```