DNS n°04 d'informatique

A rendre le mardi 13 mars.

Exponentiation rapide
Calcul de $\sqrt{2}$ avec une précision arbitraire

Exponentiation rapide

Etant donné une opération $*$ sur des objets (par exemple des entiers, des réels, …), l'objectif est de calculer $x^n = x * x * x \ldots * x$ en un nombre minimal d'opérations. On suppose que l'opération $*$ est associative, c'est-à-dire que pour tout triplet (x, y, z) on a $x * (y * z) = (x * y) * z$. Un premier algorithme naïf donne un coût en nombre d'itérations en $O(n)$.

```
PUISSANCE-NAIF(x,n)
    Données : Un objet x et un entier n positif
    Résultat : La valeur de x^n

    y = x
    for i = 2 à n faire y = y ⋆ x
    retourner y
```

1. Ecrire un code python qui met en œuvre l'algorithme ci – dessus

Pour améliorer l'algorithme naïf on utilise la propriété d'associativité de l'opération $*$. C'est-à-dire que $x^{2n} = (x^n)^2$, on calcule d'abord x^n puis par une seule opération on déduit x^{2n}. L'algorithme est donné ci – dessous.

```
PUISSANCE-DIV(x,n)
    Données : Un objet x et un entier n positif
    Résultat : La valeur de x^n
    if n = 1                                              // cas de base
        retourner x
    else                                                  // récursion
        if n pair
1           z =PUISSANCE-DIV(x,n/2)
            retourner z ⋆ z
        else                                              // n est impair ⩾ 3
2           z =PUISSANCE-DIV(x,(n − 1)/2)
            retourner z ⋆ z ⋆ x
```

2. Proposer un code python qui met en œuvre cet algorithme.
3. Faites le lien entre la séquence d'appels de la fonction $PUISSANCE - DIV$ et la décomposition en binaire de n. Le code ci – dessous peut vous aider à répondre à cette

question. Il renvoie l'écriture en binaire d'un entier sous forme d'une chaîne de caractères.

```
 8 def Dec2Bin(N) :
 9     s = ''
10     while N != 0 :
11         if N%2 == 1 :
12             s = '1' + s
13         else :
14             s = '0' + s
15         N = N//2
16     return s
17 # Test de la fonction
18 print(Dec2Bin(19))
```

4. Montrer par récurrence que la complexité de cet algorithme est en $O\big(log(n)\big)$.

On peut utiliser une méthode basée sur la décomposition en facteur de n. L'algorithme est donné ci – dessous.

```
PUISSANCE-FACT(x,n)
  Données : Un objet x et un entier n positif
  Résultat : La valeur de x^n
  if n = 1                                                  // cas de base
  |  retourner x
  else                                                      // récursion
  |  if n est premier
1 |  |  z =PUISSANCE-FACT(x,n − 1)
  |  |  retourner z ⋆ z
  |  else                                                  // n est composé n = p × q
2 |  |  y =PUISSANCE-FACT(x,p)
  |  |  z =PUISSANCE-FACT(y,q)
  |  |  retourner z
```

5. Proposer un code python qui met en œuvre cet algorithme. Vous pourrez utiliser le code ci – dessous qui renvoie deux entiers p et q les plus grands possibles tels que $n = p * q$.

```
 8 def Fact(n) :
 9     rn = int(n**0.5)
10     p = 1
11     for i in range(1,rn+1) :
12         if n%i == 0 :
13             p = i
14     return p, n//p
```

6. Comme le nombre de façons de calculer x^n est fini on peut construire l'arbre des puissances qui est donné ci – dessous pour les premières valeurs de n. En particulier pour calculer x^{31} on calcule successivement x^2, x^3, x^5, x^7, x^{14}, x^{28} et x^{31} (ce qui fait en tout 7 opérations). Par comparaison avec la décomposition en binaire on aurait eu besoin de 8 opérations et 10 opérations en factorisant. Donner pour cette exemple la suite des opérations à effectuer en utilisant la méthode de décomposition en binaire et la suite des opérations avec la méthode basée sur la décomposition en facteur.

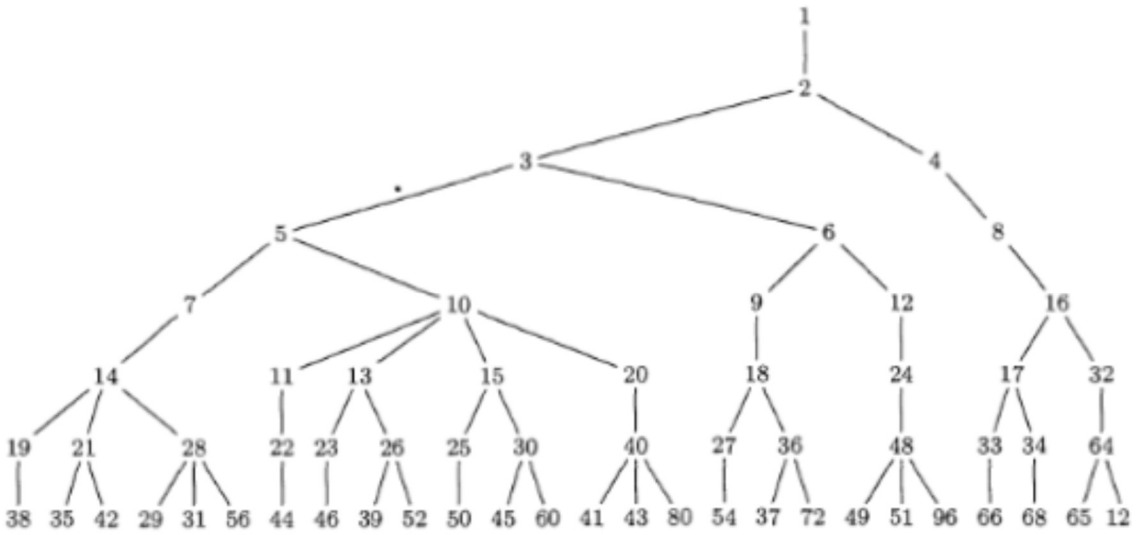

Fig. 14. The "power tree."

Calcul de $\sqrt{2}$ avec une précision arbitraire

On considère la suite définie ci – dessous qui converge très rapidement vers $\sqrt{2}$ (on double le nombre de chiffres significatifs à chaque itération (vous avez peut – être reconnu la méthode de Newton).

$$x_{n+1} = \frac{x_n}{2} + \frac{1}{x_n}$$

1. Proposer un code qui permet de calculer les éléments de cette suite. Vérifier que ça converge rapidement vers $\sqrt{2}$

En démarrant cette suite avec un entier on peut se convaincre rapidement que tous les éléments de cette suite vont être des nombres rationnels. On note pour cela les éléments de la suite x_n sous la forme $x_n = \frac{N_n}{D_n}$.

2. Etablir la relation de récurrence pour les suites N_n et D_n.
3. Ecrire une fonction qui renvoie les valeurs de N_{n+1} et D_{n+1} à partir des valeurs de N_n et D_n.
4. Ecrire une fonction qui renvoie la valeur de la fraction $\frac{N_n}{D_n}$.en écriture décimale (comme vous le faisiez en primaire pour obtenir l'écriture décimale dans une division de deux entiers).
5. Donner alors la valeur de $\sqrt{2}$ avec 1000 chiffres significatifs (une douzaine d'itérations suffisent).
6. $\sqrt{2}$ est difficilement approchable par une fraction rationnelle. Cela s'écrit mathématiquement sous la forme ci – dessous. Vérifier que cette relation est bien vérifiée.

$$\left|\sqrt{2} - \frac{p_n}{q_n}\right| \sim \frac{1}{{q_n}^2}$$