Fiche 8- Proposition de solutions

Solution 1 La taille binaire de N est de l'ordre de $\log_2(N)$: c'est le nombre chiffre de son écriture binaire.

- l'opération somme consiste à parcourir de droite à gauche les deux nombres et à sommer les bits correspondant, en tenant compte d'une retenue éventuelle : $\mathcal{O}(\ln(N))$
- un test d'égalité ou une comparaison consiste à détecter la première différence de bits, en partant de la droite, dans les écritures des deux nombres : $\mathcal{O}(\ln(N))$
- le produit peut être considérer comme la répétition de l'addition, $\mathcal{O}(\log(N)^2)$, ou se faire avec des algorithmes plus efficace en $\mathcal{O}(\log(N)\log(\log(N)))$.

Solution 2 1. `Algo A` : nombre de d'itérations :

$$\sum_{i=0}^{n} i+1 = \frac{(n+1)(n+2)}{2}$$

La complexité est donc : $\mathcal{O}(n^2)$.
2. `Algo B` : il y a n itérations : la variable `i` est incrémentée de 1 à n. La complexité est $\mathcal{O}(n)$.
3. `Algo C` :
a) la variable `i` est incrémentée de 1 à chaque boucle : elle peut servir pour repérer les boucles.
b) lors de la boucle de rang `i`, la variable `j` contient :

$$\sum_{k=0}^{i} k = \frac{i(i+1)}{2}$$

c) le test d'arrêt est `j` $\geq$ `n` c'est-à-dire : on cherche `i` tel que

$$\frac{i(i+1)}{2} \geq n$$

Ainsi, la boucle s'arrête dès que $i \approx \sqrt{2n}$.
La complexité est donc $\mathcal{O}(\sqrt{n})$.

Solution 3 1. La complexité de `diviseurs` est $\mathcal{O}(n)$.
2. Une amélioration est :

```
def diviseurs2(n):
    k=1
    while k*k<n:
        if n%k==0:
            print(k,n//k)
        k=k+1
    if k*k==n:
        print(k)
```

La boucle contient $\lfloor \sqrt{n} \rfloor$ itérations. Le calcul de d est du même ordre. Ainsi, la complexité de `diviseurs2` est $\mathcal{O}(\sqrt{n})$.

Solution 4 1. Algorithme complété :

```
s ← 1
m ← x
f ← 1
pour k variant de 1 à n faire
  s ← s+m/f
  m ← m*x
  f ← f*(k+1)
fintantque
afficher s
```

2. L'invariant de boucle est, pour $k \in [\![1, n]\!]$:

$$\begin{cases} \mathtt{s} = \sum_{j=0}^{k-1} \frac{x^j}{j!} \\ \mathtt{m} = x^k \\ \mathtt{f} = k! \end{cases}$$

3. La variable s contient au début du tour k=n : $\sum_{j=0}^{n-1} \frac{x^j}{j!}$
A la fin de ce tour, elle contient : $\sum_{j=0}^{n} \frac{x^j}{j!}$
La correction de cet algorithme est établie !
4. La suite $(n-k)$ est strictement décroissante dans $\mathbb{N}$. Il y a exactement n itérations. L'algorithme se termine.
5. Nouvelle version de l'algorithme :

```
s ← 1
m ← 1
f ← 1
pour k variant de 1 à n faire
  m ← m*x
  f ← f*k
  s ← s+m/f
fintantque
afficher s
```