TD 2 : Terminaison, correction et complexité

Exponentiation rapide

On considère le code suivant :

```
def exp(a,n) :
    r = 1
    while n > 0 :
        if n % 2 == 0 :
            n = n // 2
        else :
            n = (n - 1) // 2
            r = r * a
        a = a*a
    return r
```

1. Tester la fonction `exp` sur les couples (2,4) et (3,3) en faisant apparaître l'évolution des paramètres en cours d'exécution.
   Quel est le rôle de cette fonction ?
2. Justifier la terminaison de cet algorithme.
3. Si $n = \sum_{i=0}^{p} \alpha_i 2^i$ avec $\alpha_i \in \{0,1\}$ est la décomposition en base 2 de l'entier n, on introduit les suites (a_k), (n_k) et (r_k) qui contiennent les valeurs de `a`, `n` et `r` à l'issue de la $k^{\text{ième}}$ boucle.
   Déterminer la valeur de (a_k), (n_k) et (r_k) en fonction de k et de la décomposition de n.
4. Prouver la correction de cet algorithme.

Nombre de boucles

Déterminer la complexité des algorithmes suivants (nombre de boucles) :

1.
```
x = 0
for i in range(n) :
    for j in range(n) :
        x += 1
```

2.
```
x = 0
for i in range(n):
    for j in range(i):
        x += 1
```

3.
```
i,x = n,0
while i > 1:
    x += 1
    i //= 2
```

4.
```
i,x = n,0
while i > 1:
    for j in range(i):
        x += 1
    i //= 2
```

Complexité d'algorithmes

Déterminer la complexité dans le meilleur des cas, dans le pire des cas pour les fonctions Python suivantes, après avoir déterminé leur but :

1.
```
def fonction1(L):
    n = len(L)
    for i in range(n-1,1,-1) :
        for j in range(i) :
            if L[j] > L[j+1] :
                L[j],L[j+1] = L[j+1],
                    L[j]
    return L
```

On pourra tester cet algorithme sur la liste $[2,4,1]$ pour déterminer son but.

2.
```
def fonction2(n):
    p = 0
    while n%2 == 0 :
        n = n//2
        p = p+1
    if n == 1 :
        a = p
    else :
        a = 0
    L = [a,p+1]
    return L
```