Feuille d'exercices : Complexité et preuves d'algorithmes

Exercice 1 Dans cet exercice `truc` et `bidule` désignent deux instructions et n un entier naturel. Déterminer en fonction de n pour chaque script le nombre de fois où les instructions `truc` et `bidule` sont exécutées.

```
# script 1
for i in range(n):
    truc
for j in range(n):
    bidule
```

```
# script 2
for i in range(n):
    truc
    for j in range(n):
        bidule
```

```
# script 3
for i in range(n):
    truc
    for j in range(i,n):
        bidule
```

Exercice 2 Déterminer pour chacun des scripts le nombre d'opérations significatives effectuées, en déduire leur complexité en fonction de n.

```
# script 1
n = 100
s = 0
for i in range(n):
    for j in range(n):
        for k in range(n):
            s = 2 + s
```

```
# script 2
n = 100
s = 0
for i in range(n):
    s = s + 3
for j in range(n):
    s = s + 3
```

```
# script 3
n = 20
s = 0
for i in range(n):
    for j in range(i,n):
        s = s + j**2
```

```
# script 4
n = 100
p = 1
for i in range(n):
    for j in range(i-5,i+5):
        p = 2*p
```

```
# script 5
n = 100
s = 0
for i in range(n):
    a = n
    while a > 1:
        a = a/2
    s += 1
```

```
# script 6
n = 20
i = n
s = 0
while i > 1:
    for j in range(i):
        s = s + 1
    i = i//2
```

Exercice 3 (Terminaison et complexité) Étudier la terminaison des scripts suivants et déterminer le cas échéant le nombre d'itérations.

```
# script 1
n = 100
a = 1
while a < n:
    a = a + 5
```

```
# script 2
n = 10**8
a = 1
while a < n:
    a = 2 * a
```

```
a = 100
while a >= 0:
    a = a//2
```

Exercice 4 (Invariant de boucle) Justifier la terminaison puis démontrer que la fonction suivante renvoie le produit de a par b en utilisant l'invariant de boucle $p = i \times b$.

```
def multiplication(a,b):
    p = 0
    i = 0
    while i < a:
        p = p + b
        i = i + 1
    return p
```

Exercice 5 (Invariant de boucle) Démontrer que la propriété suivante «à l'entrée de la boucle i, la variable `maximum` est égale à $\max\{t[0], t[1], \dots, t[i-1]\}$» est un invariant de boucle. En déduire que `maxi(t)` renvoie bien le maximum des éléments de `t`.

```
def maxi(t):
    """Données: t un tableau d'entiers
       Résultat: le maximum des éléments de t"""
    n =len(t)
    if n == 0:
        return None # cas où le tableau est vide
    maximum = t[0]
    for k in range(1,n):
        if t[k] >  maximum:
            maximum = t[k]
    return maximum
```

Exercice 6 (Une tentative primale) On considère la suite d'entiers $(p_n)_n$ définie par $p_0 = 2$ et :

$$\forall n \in \mathbb{N}^*, p_n = p_0 \times p_1 \times \cdots \times p_{n-1} + 1.$$

Par exemple, $p_1 = p_0 + 1 = 3$, $p_2 = p_0 \times p_1 + 1 = 7$.

1. On prend $n = 10$. Contruire un tableau t de longueur $n+1$ tel pour tout entier k entre 0 et n, $t[k]$ contient la valeur de p_k. En déduire une fonction `suite(n)` qui prend en argument n un entier naturel et renvoie le nombre p_n. Par exemple, l'exécution de `suite(2)` renvoie le nombre 7.
2. Déterminer la complexité temporelle et spatiale de la fonction `suite(n)`.

On considère la fonction `truc(n)` suivante.

```
def truc(n):
    p = 2
    q = 1
    for k in range(n):
        q  = q*p
        p = q + 1
    return p
```

3. On pose pour tout $n \in \mathbb{N}$, $q_n = p_n - 1$. Démontrer que la propriété «en sortie de boucle n°k, la variable `p` est égale à p_k et la variable `q` est égale à q_k» est un invariant de boucle. En déduire que `truc(n)` renvoie bien l'entier p_n.
4. Déterminer la complexité temporelle et spatiale de la fonction `truc(n)`.