TD 2 : Complexité et analyse des algorithmes

Exercice 1 - Notations asymptotiques

Question 1

Montrer que si $f(n) = (n + 1)^2$ alors f(n) a $O(n^2)$.

Question 2

Montrer que si $f(n) = 3(n^3) + 2(n^2)$ alors f(n) a $O(n^3)$.

Question 3

Montrer que si $f(n) = 3(n^3) + 2(n^2)$ alors f(n) a $\Omega(n^3)$.

Question 4

Montrer que si $f(n) = 3^n$ alors f(n) ´ $O(2^n)$.

Raisonnement par récurrence - Rappel

Soit P une propriété sur les entiers.

Lorsqu'on raisonne par récurrence on peut être amené à utiliser les deux schémas de récurrence suivants:

Schéma 1 :

Si les deux propositions suivantes sont vérifiées:

- $P(b)$
- $P(n) \rightarrow P(n+1)$

Alors, pour tout entier n tel que $b \leq n$, $P(n)$ est vraie.

Schéma 2 :

Si les deux propositions suivantes sont vérifiées :

- $P(b)$
- (Pour tout entier $b \leq k \leq n$, $P(k)) \rightarrow P(n+1)$

Alors, pour tout entier n tel que $b \leq n$, $P(n)$ est vraie.

Exercice 2 – Preuve et complexité d'algorithmes

1. Soit l'algorithme récursif F suivant, avec un paramètre entier n :

```
def F(n):
#Input: n un entier
#Output:?
  if n==0:
    return 2
  else:
    return F(n-1)*F(n-1)
```

Que calcule cet algorithme ? Prouver l'algorithme.

2. Déterminer la complexité de l'algorithme F.

3. Soit l'algorithme itératif G suivant, avec un paramètre entier *n* :

```
def G(n):
#Input: n un entier
#Output:?
  r = 2
  for i in range(n+1):
    r = r*r
  return r
```

Que calcule cet algorithme ? Prouver l'algorithme.

4. Déterminer la complexité de l'algorithme G.

Exercice 3 - Complexité en fonction de deux paramètres

Déterminer la complexité des algorithmes suivants en fonction de *m* et *n* (préalablement initialisés) par rapport au nombre d'itérations effectuées.

```
def A(m,n):
#Input:
#  m un entier
#  n un entier
   i = 1
   j = 1
   while (i ≤ m) and (j ≤ n):
      i = i+1
      j = j+1
```

```
def B(m,n):
#Input:
#  m un entier
#  n un entier
   i = 1
   j = 1
   while (i≤ m) or (j ≤ n):
      i = i+1
      j = j+1
```

```
def C(m,n):
#Input:
#  m un entier
#  n un entier
   i = 1
   j = 1
   while (j ≤ n):
      if (i ≤ m):
         i = i+1
      else:
         j = j+1
```

```
def D(m,n):
#Input:
#  m un entier
#  n un entier
   i = 1
   j = 1
   while (j ≤ n):
      if (i ≤ m):
         i = i+1
      else:
         j = j+1
         i = 1
```

Exercice 4

Déterminer la complexité des algorithmes suivants en fonction de la variable entière *n* qui a été initialisée, en comptabilisant toutes les opérations (affectations, opérations arithmétiques, tests) :

a)
```
s = 0
i = 0
while (i ≤ n)
   j = 0
   while (j ≤ n):
      s = s+1
      j = j+1
   i = i+1
```

b)
```
s = 0
i = 0
while (i ≤ n):
   j = 0
   while (j ≤ n*n):
      s = s+1
      j = j+1
   i = i+1
```

c)
```
s = 0
i = 0
while (i ≤ n)
   j = 0
   while (j ≤ i):
      s = s+1
   i = i+1
```

d)
```
s = 0
i = 0
while (i ≤ n)
   j = 0
   while (j ≤ i*i):
      k = 0
      while k ≤ j :
         s = s+1
         k = k+1
      j= j+1
   i = i+1
```

Que devient la complexité de l'algorithme d) si l'on rajoute j = j+1 dans le corps de la seconde boucle ?

a)

b)

c)

d)

Exercice 5 : (traité en cours) - Preuve d'algorithmes et calcul de leur complexité

Question 1

Concevons un algorithme pour déterminer si au moins une personne dans l'auditoire vient de la même ville que vous.

Donner la complexité (nombre de questions à poser) de chaque algorithme dans le pire des cas, en fonction du nombre d'étudiants dans l'auditoire.

Algorithme 1

Vous dites à un premier camarade d'où vous venez, et lui demandez s'il vient de cette ville également. S'il dit non, vous faites de même avec un second camarade, etc.

Algorithme 2

Vous dites à tout le monde d'où vous venez et demandez à l'auditoire s'il y a quelqu'un qui vient également de là.

Algorithme 3

Chaque personne ne peut parler qu'à une seule autre. Vous dites donc à un premier camarade (étudiant 1) d'où vous venez, et lui demandez s'il vient de là. Si non, vous lui dites de demander à son voisin (étudiant 2). Il vous retransmet la réponse de l'étudiant 2. Si non, vous lui dites de faire transmettre la question à l'étudiant 3, etc.

Algorithme 4

Chaque personne veut savoir si dans la salle il y a quelqu'un qui vient de la même ville qu'elle.