Série 5 de Travaux Dirigés

Objectifs :	Analyser la complexité des algorithmes.

Exercice 1.

En considérant le pire des cas où le tableau est initialement trié dans le sens inverse (décroissant), analysez les complexités des algorithmes de tri étudiés dans la série TD4 : le tri par sélection, par insertion et à bulles.

Exercice 2.

On considère le code suivant, comportant deux « tant que » imbriqués. On cherche à estimer la complexité de cette imbrication en fonction de n.

Pour cela, on utilise la variable compteur, qui est incrémentée à chaque passage dans le « tant que » interne.

```
fonction f(n : entier ) :entier
var compteur, i,j : entiers ;
début
        compteur ← 0 ; i ← 1
        tant que (i < n) faire
                j ← i + 1
                tant que (j <= n) faire
                        compteur ← compteur + 1
                        j ← j + 1 ;
                finTantque_j
                i ← i * 2
        finTantque_i
        retourner compteur
fin
```

1. Quelle est la valeur finale du compteur dans le cas où n = 16 ?
2. Considérons le cas particulier où n est une puissance de 2 : on suppose que $n = 2^p$ avec p connu. Quelle est la valeur finale du compteur en fonction de p ? Justifiez votre réponse.
3. . Exprimez le résultat précédent en fonction de n. d. En conclure la complexité dans le pire des cas, en notation O, de cette fonction.

Exercice 3.

Les coefficients du binôme sont définis par la récurrence :

$C_n^0 = 1 \quad C_n^n = 1$

$C_n^k = C_{n-1}^k + C_{n-1}^{k-1} \text{ pour } 0 < k < n$

Pour minimiser le coût on peut utiliser une technique de programmation dynamique qui consiste à stocker les valeurs intermédiaires pour ne pas les recalculer : c'est le **triangle de Pascal.**

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    1   6   15  20  15  6   1
  1   7   21  35  35  21  7   1
1   8   28  56  70  56  28  8   1
```

.

1. Ecrire l'algorithme du triangle de pascal pour calculer
2. Montrer que la complexité de cet algorithme est de l'ordre de $O(n^2)$