Factorielle

```
fonction factorielle(n)
  si n vaut 0 alors
    retourner 1
  sinon
    retourner n x factorielle(n-1)
  fin si
```

Fibonacci

```
fonction F(n) (version récursive)
  si n vaut 0 ou 1 alors
    retourner n
  sinon
    retourner F(n-1) + F(n-2)
  fin si
```

Tours de Hanoï

```
Transférer n disques du piquet O (origine) vers le
piquet D (destination) via le piquet I (intermédiaire):
  si n > 0 alors
    Transférer n-1 disques de O vers I via D
    transférer le disque restant de O vers D
    Transférer n-1 disques de I vers D via O
  fin si
```

Permutation

```
fonction permuter(S,n)
  si n vaut 1
    S est la seule permutation
  sinon
    pour toutes les lettres c de S, boucler
      permuter(S,n-1)
      Placer c en dernière position
   fin boucler
  fin si
```

Heap (permutation)

```
fonction permuter(S,n)
  si n vaut 1
    traiter(S)
  sinon
    permuter(S,n-1)
    pour i allant de 1 à n-1, boucler
      si n est pair
        échanger S(i) et S(n)
      sinon
        échanger S(1) et S(n)
      fin si
      permuter(S,n-1)
    fin boucler
  fin si
```

MinMax

```
fonction calculeScore( case, joueur )
   marquer la case
   si la grille est gagnante pour joueur, alors
      score ← +1
   sinon si la grille est pleine, alors
      score ← 0
   sinon
      scoreAdverse ← -∞
      pour toute case vide c
         scoreAdverse ← max(scoreAdverse,
                            calculeScore(c,adversaire))
      fin pour
      score ← -1 * scoreAdverse
   fin si
   effacer la marque dans la case
   retourner score
```

	Itératif	Récursif
Factorielle	O(n)	
Fibonacci	O(n)	$O(1.618^n)$
PGCD (Euclide)	O(log(n))	
Hanoï	$O(2^n)$	
Permutations	O(n!)	
Tic Tac Toe	9!	
Puissance 4, profondeur d'exploration de d tours	$O(7^d)$	
Minimax, m mouvements possibles par tour, profondeur de d tours	$O(m^d)$	