Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Análisis de algoritmos - Ejercicios Parcial 1

1. Análisis asintótico

Ordene las siguientes expresiones de menor a mayor orden de crecimiento

- $8n^2$
- $n\log_6 n$
- 64
- $\log_2 n$
- $n\log_2 n$
- $8n^2$
- $6n^3$
- $\log_8 n$
- 8^{2n}

2. Invariante de ciclo

Para el siguiente algoritmo, identifique el invariante del algoritmo POW

Algoritmo 1 Algoritmo POW.

```
1: procedure POW(b, e)
Require: b, e ∈ ℕ ∧ b, e > 0: el algoritmo calcula b^e
2:     r ← b
3:     for i ← 2 to e do
4:         r ← r × b
5:     end for
6:     return r
7: end procedure
```

1. Encuentre el invariante del ciclo del algoritmo POW y su demostración
   a) Condición del invariante
   b) Inicio
   c) Mantenimiento
   d) Terminación

3. Invariante de recursión

Para el siguiente algoritmo, identifique el invariante del algoritmo POW_REC

1. Encuentre el invariante de la recursión del algoritmo POW_REC y su demostración
   a) Condición del invariante
   b) Inicio
   c) Mantenimiento
   d) Terminación

Algoritmo 2 Algoritmo POW_REC.

```
1: procedure POW_REC(b, e)
Require: b, e ∈ ℕ ∧ b, e > 0: el algoritmo calcula b^e
2:     if e = 1 then
3:         return b
4:     else
5:         r ← ⌊e ÷ 2⌋
6:         v ← POW_REC(b, r)
7:         if e mod 2 = 0 then
8:             return v × v
9:         else
10:            return v × v × b
11:        end if
12:    end if
13: end procedure
```

4. Análisis de un algoritmo dividir-y-vencer

Considere el algoritmo:

Algoritmo 3 Algoritmo Foo.

```
Require: S = ⟨s_i ∈ ℕ⟩ ∧ p ≤ r: p y r representan los límites de la secuencia.
1: procedure Foo(S, p, r, b)
2:     q ← ⌊(p + r) ÷ 2⌋
3:     if p > r then
4:         return −1
5:     else if S[q] = b then
6:         return q
7:     else
8:         x ← Foo (S, p, q, b)
9:         y ← Foo (S, q + 1, r, b)
10:        return máx (x, y)
11:    end if
12: end procedure
```

1. Calcule el orden de complejidad del algoritmo.
2. ¿Qué hace el algoritmo?
3. Escriba una versión iterativa del algoritmo.

5. Escritura de un algoritmo

Para este ejercicio se define el problema de contar cuántos valores invertidos tiene una secuencia $A = \langle A_i \in \mathbb{N} \rangle$.
Un valor invertido en una secuencia A es un par de índices (i, j) tal que:
$i < |A| \wedge j \leq |A| \wedge j > i \wedge A[i] > A[j]$
Cree un algoritmo para contar cuántos valores invertidos tiene la secuencia A

1. Análisis del problema.
2. Diseño del algoritmo.
3. Pseudocódigo del algoritmo.
4. Análisis de complejidad.