Notación asintótica y conteo de instrucciones
ADA - Práctica

López del Álamo, Christian
Asistente: Guillén Dávila, Carlos
Asistente: Guzmán Rodríguez, Renato

1. Ordenar las siguientes por tasa de crecimiento asintótico en forma ascendente

$lg(lg^* n)$	$2^{lg^* n}$	$\sqrt{2}^{lg(n)}$	n^2	$n!$	$(lg\, n)!$
$\left(\frac{3}{2}\right)^n$	n^3	$lg^2 n$	$lg(n!)$	2^{2^n}	$n^{1/lg\, n}$
$\ln\ln n$	$lg^* n$	$n \cdot 2^n$	$n^{\lg\lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2\lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

2. Para la siguiente tabla decidir si f(n) = O(g(n)) o si g(n) = O(f(n)) (justificar):

$f(n)$	$g(n)$
$10n$	$\sqrt{n^2+5}$
n^3	$n^2 \log^3 n$
$n \log_2 n$	$n + \ln(n^5)$
$5^{\log_2 n}$	$5^{\ln n}$
$\log(\log n)$	$\ln(\sqrt{n})$
2^n	16^n
2^{n^2}	n^{2^n}
$\cos n$	$\sin n$
n^2	$(n \cos n)^2$

3. Decidir si los siguientes enunciados son verdaderos o falsos (justificar):

1. Sean f y g funciones asintóticamente no negativas. Entonces al menos alguna de las relaciones: $f = O(g)$, $g = O(f)$ deberá cumplirse siempre.
2. Para cualquier par de funciones positivas f y g, Si $g(n) = O(n)$ entonces $f(g(n)) = O(f(n))$
3. Sean $f(n) = \sqrt{n^e}$ y $g(n) = n \,.\, (n \,mod\, 100)$. Entonces $f(n) = O(g(n))$.
4. Sean $f(n)$y $g(n)$ dos funciones asintóticamente positivas. Si $f(n) = \Theta(n)$ entonces $|f(n) - g(n)| = \Theta(f(n) + g(n))$
5. Si f tiene cota superior polinómica. Entonces se cumple $\log(f(n)) = O(\log n)$
6. La función $\lceil \log n \rceil !$ tiene cota superior polinómica.
7. La función $\lceil \log\log n! \rceil$ tiene cota superior polinómica
8. Sea F_k el kth número de Fibonacci computado en un tiempo $O(\log n)$. Entonces el n^2th número de Fibonacci F_{k^2} es computado en tiempo $O(\log^2 n)$.

4. Probar las siguientes sumatorias:

1. $\sum_{i=0}^{n} i = O(n^2)$
2. $\sum_{i=0}^{n} i^2 = O(n^3)$
3. $\sum_{i=0}^{n} \frac{1}{i} = O(\log n)$
4. $\sum_{i=0}^{n} a^i = O(1)$, donde $0 \leq a < 1$

5. Conteo de instrucciones

1. Analiza y realiza el conteo de instrucciones del siguiente código. Indica a qué algoritmo se hace referencia.

```
void fun (vector <int> &V) {
    for (int i = 0; i < V.size(); i++)
        for (int j = i+1; j < V.size(); j++)
            if (V[i] > V[j])
                swap (V[i], V[j]);
}
```

2. Analiza y realiza el conteo de instrucciones del siguiente código.

```
int fun (int n, int k) {
        int d = 0;
    while(n/k)  {
        d += n / k;
        n = n / k + n %k;
    }
    return d;
}
```

3. Analiza y realiza el conteo de instrucciones. Indica qué retorna.

```
pair<string, int> fun (int n) {
    string r;
    while (n) {
        r = char((n & 1)+'0')+r;
        n >>= 1;
    }
    return r;
}
```

4. Analiza y realiza el conteo de instrucciones del siguiente código e indica qué hace.

```
int fun(vector<int> &V) {
        set<int> C;
        int n = V.size();
        while(n--)
        C[V[n]]++;
        return C.size();
}
```

5. Analiza y realiza el conteo de instrucciones del siguiente código e indica a qué algoritmo se hace referencia.

```
void fun(vector <int> &V) {
    for (int i = 1; i < V.size(); ++i) {
        int index = V[i], j;
        for (j = i-1; j >= 0 && V[j] > index; --j)
           V[j+1] = V[j];
        V[j+1] = index;
        }
}
```

6. Analiza y realiza el conteo de instrucciones del siguiente código. Qué hace la función?

```
void fun(vector <int> &V) {
        int n = *max_element(V.begin(), V.end());
        vector<int> C(n+1);

        for(int i = 0; i < V.size(); ++i) {
                ++C[V[i]];
        }

        for(int i = 0, j = 0; i < C.size(); ++i) {
                while(C[i]--) {
                        V[j] = i;
                        ++j;
                }
        }
}
```

7. Analiza y realiza el conteo de instrucciones. Intenta averiguar que estructura retorna (Teoría de números (primos)).

```
vector<bool> fun (int n) {
        vector <bool> V(n, 1);
    V[1] = 1;
    for(int i = 2; i*i < V.size(); ++i)
        if(V[i])
            for(int j = i+i; j < V.size(); j +=i)
                V[j] = 0;
    }
        return V;
}
```

8. Sea B una base y P su exponente. Analiza y realiza el conteo de instrucciones. Averigua que retorna.

```
long long fun(long long B, long long P, long long M) {
    long long r = 1;
    while (P > 0) {
        if (P & 1) {
            r = (r * B) %M;
        }
        B = (B * B) %M;
        P >>= 1;
    }
    return r;
}
```

9. Analiza y realiza el conteo de instrucciones. ¿Qué figura se grafica?

```
void fun(int a, int f) {
        for (int j = 0; j < f; ++j)     {
                if (j) cout << endl;
                for (int k = 0; k < a; ++k) {
                        for (int l = 0; l < k+1; ++l)
                                cout << (k %a) + 1;
                        cout << endl;
                }

                for (int k = 0; k < a-1; ++k) {
                        for (int l = 0; l < a-1-k; ++l)
                                cout << a-1-k;
                        cout << endl;
        }
    }
}
```

10. Sea M una matriz triangular superior de la forma:

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

Analiza y realiza el conteo de instrucciones.

```
int fun(vector<vector<int> > &M) {
        int m = M.size();
        int n = M[0].size();

        vector<vector<int> > AC = vector<vector<int> > (m, vector<int> (n));

        for(int i = 0; i < m; ++i)
        for(int j = 0; j < n; ++j)
            AC[i][j] = M[i][j] ? i ? (AC[i-1][j] + 1) : 1 : 0;

        int accumulator = 0, k;
        for(int i = 0; i < m; ++i) {
                for(int j = 0; j < n; ++j) {
                        k = j;
                        int maxprevious = AC[i][k];
                        while(AC[i][k] && k+1) {
                                accumulator += maxprevious;
                                k--;
                                if(maxprevious > AC[i][k])
                                        maxprevious = AC[i][k];
                        }
                }
        }
        return accumulator;
}
```