Boletín de Ejercicios Tema 2

Programación Avanzada

Grado en Ingeniería Informática y Tecnologías Virtuales

Programación III

Grado en Ingeniería del Software

Curso académico 2025/2026

Antonio M. Durán Rosal

Aspectos Generales (Ejercicios 1-20)

Para cada ejercicio se pide:

- Definir formalmente la función recursiva no final y final.
- Implementar la función recursiva no final (si lo considera necesario) y final.
- Definir e implementar la solución iterativa.
- Calcular $T(n)$ y su orden de complejidad.

Hay que mirar en la rúbrica disponible en Moodle que partes específicas en cada ejercicio se evaluarán.

Nivel Básico

Ejercicio 1. Dado un array de enteros, devolver el producto de todos sus elementos. Si el array está vacío se devuelve 1 por conveniencia.

Ejercicio 2. Dado un array de `float`, suma el cuadrado de la raíz cúbica de sus elementos.

Ejercicio 3. Dada una cadena de caracteres, contar cuántas vocales contiene.

Ejercicio 4. Dado un array, decidir si todos los elementos son cuadrados perfectos.

Ejercicio 5. Dado un número entero n, calcular el valor de la suma $1 + 2 + 3 + ... + n$.

Ejercicio 6. Dados dos número a y b (ambos positivos mayores o iguales que cero), obtener la multiplicación por las sumas sucesivas.

Ejercicio 7. Dado un array de booleanos, determinar si todos sus elementos son `true`.

Nivel Medio

Ejercicio 8. Dado un array de cadenas, encontrar la cadena con mayor longitud (la primera del vector en caso de empate).

Ejercicio 9. Dado un array, decidir si todos los elementos son números perfectos (se puede hacer uso de una función auxiliar que compruebe si un elemento es un cuadrado perfecto o no, y cuya complejidad sea lineal).

Ejercicio 10. Dada una cadena de caracteres, decidir si es un palíndromo. Una cadena es un palíndromo si es igual a su inversa. Por ejemplo: ana, arenera, aviva, radar, reconocer, salas, ...

Ejercicio 11. Dado un array de tipo `Vuelo` (definido por el `string` nombre, `int` asientos, y `double` precio) buscar el vuelo con el mayor ganancia (en caso de empate, el que vaya antes en el vector de izquierda a derecha).

Ejercicio 12. Dada una base b y un exponente e, generar la suma de las potencias de b desde b^1 hasta b^e (usar la función `pow` suponiendo que su complejidad es lineal).

Ejercicio 13. Defina de manera recursiva la combinatoria con repeticiones (Pista: usar la definición recursiva sin repeticiones disponible en la teoría).

Ejercicio 14. Realizar la transformación a iterativo de la siguiente función recursiva definida para $n >= 0$:

$$f(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 3f(n-1) + f(n-2) & \text{si } n > 1 \end{cases} \tag{1}$$

Nivel Avanzado

Ejercicio 15. Encontrar el espejo de la parte derecha de una cadena de caracteres. Por ejemplo, dada la palabra “antonio”, el resultado sería “oinonio”.

Ejercicio 16. Dado un entero, obtener una lista de sus divisores primos (se puede implementar una función que determine si un número es primo o no, y cuya complejidad sea lineal). Consideraremos el 1 como número primo.

Ejercicio 17. Dado un número, determinar el producto, el número de cifras pares y la suma de sus cifras. Por ejemplo: dado 5394, el resultado debe ser {540, 1, 21}.

Ejercicio 18. Dado un número entero, obtener la suma de los cuadrados de las cifras pares, la cantidad de cifras impares, y el valor mínimo de sus cifras. Por ejemplo: dado 5863, el resultado sería {100, 2, 3}.

Ejercicio 19. Realice la División por restas sucesivas. Si se desea realizar A/B y obtener el resultado S y el resto R, el proceso consistirá en restar a A la cantidad B hasta que el resultado de la resta sea menor que el propio B. S será el número de operaciones de resta realizadas y R el resultado de la última resta.

Ejercicio 20. Realizar la transformación a iterativo de la siguiente función recursiva definida para $n >= 0$:

$$g(n) = \begin{cases} n^2 & \text{si } n < 3 \\ 2g(n-1) - g(n-2) + g(n-3) & \text{si } n \geq 3 \end{cases} \quad (2)$$

Aspectos Generales (Ejercicios 21-30)

Para cada ejercicio se pide:

- El tamaño del problema n.
- Cálculo de $T(n)$.
- Orden de complejidad.

Nivel Variable

Ejercicio 21.

```
int i = 1;
while (i < n)
{
  i = i * 3;
  for (int j = 0; j < n; j++)
  {
    x++;
  }
}
```

Ejercicio 22. Suponiendo $C(n) \in O(n^2)$:

```
for (int i = 0; i < n; i++)
{
  if (i % 2 == 0)
  {
    r += C(i);
  }
}
```

Ejercicio 23.

```
for (int i=n; i<=n-1; i++)
{
  for(int j=1; j<=n; j++)
  {
    for(int k=1; k<=n; k++)
    {
      r = r + 2;
    }
  }
}
```

Ejercicio 24. Suponiendo $a[i] \in O(n \log n)$:

```
r = 0;
for (int i=0; i<n-k; i++)
{
  if(a[i] == k)
  {
    for(j=1; j<=a[i]; j++)
    {
      r+= a[i + j];
    }
  }
}
```

Ejercicio 25. Suponiendo $a[i] \in O(n)$:

```
int busca(int [] a, int n, int c)
{
  int i = 0;
  int j = n;
  int k, r;
  boolean fin = false;
  while (j > i && !fin)
  {
    k = (i + j) / 2;
    if (a[k] == c)
    {
      r = k;
      fin = true;
    }
    else if (c < a[k])
    {
      j = k - 1;
    }
    else
    {
      i = k + 1;
    }

  }
  if(!fin)
  {
    r = -1;
  }
  return r;
}
```

Ejercicio 26. Suponiendo $a[i] \in O(1)$, determine la complejidad de f0:

```
int f(int [] a, int n, int i, int j)
{
  int k, t, s;
  if (j-i == 0)
  {
    s=0;
  }
  else if(j-i == 1)
  {
    s=a[i];
  }
  else if(j-i == 2)
  {
    s=a[i]+a[i+1];
  }
  else
  {
    k = (i + j)/2;
    t = (j - i)/3;
    s = a[k] + f(a, n, i, i+t) + f(a, n, j-t, j);
  }
  return s;
}
int f0(int [] a, int n)
{
  return f(a, n, 0, n);
}
```

Ejercicio 27.

```
int g(int n)
{
  if (n <= 1)
  {
    return 1;
  }
  else
  {
    return 2 * g(n - 1) + g(n - 2);
  }
}
```

Ejercicio 28.

```
int h(int n)
{
  if (n <= 2)
  return n;
  else
  return h(n - 1) + h(n - 3);
}
```

Ejercicio 29.

```
double f(int n, double a){
  double r;
  if (n == 1)
  {
    r = a * 2;
  }
  else
  {
    r = 3 * f(n / 2, a + 2) - f(n / 3, a - 2);
    for (int i = 1; i < n; i++)
    {
      r -= a / i;
    }
  }
  return r;
}
```

Ejercicio 30.

```
int f(int a, int b)
{
  int r;
  if (a >= tope || b >= tope)
  {
    r = 1;
  }
  else
  {
    r = 3 * f(a + 1, b + 1);
  }
  return r;
}
```