Resolución de Tarea 3 - Divide y Vencerás (Fecha: 16 de Octubre de 2025)

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI5651 - Diseño de Algoritmos I
Septiembre - Diciembre 2025
Estudiante: Junior Miguel Lara Torres (17-10303)

Tarea 3 (9 puntos)

Indice

Pregunta 1

Teniendo en cuenta la siguiente version simplificada del **Teorema Maestro**

$$\text{Para } T(n) = aT(\frac{n}{b}) + g(n)$$

$$T(n) = \begin{cases} \Theta(n^k) & \text{si } a < b^k \\ \Theta(n^k log(n)) & \text{si } a = b^k \\ \Theta(n^{log_b(a)}) & \text{si } a > b^k \end{cases}$$

Se tiene para los siguientes problemas que

- $T(n) = 3T(\frac{n}{4}) + \frac{7(n^2-1)}{3}$

  Tenemos $g(n) = \frac{7(n^2-1)}{3} \in O(n^2)$, por lo tanto

  $$\begin{matrix} a = 3 \\ b = 4 \\ k = 2 \end{matrix} \implies 3 < 4^2 \implies T(n) \in \Theta(n^2)$$

- $T(n) = 5T(\frac{n}{5}) + 7n - 4$

Tenemos $g(n) = 7n - 4 \in O(n)$, por lo tanto

$$\begin{array}{l} a = 5 \\ b = 5 \\ k = 1 \end{array} \implies 5 = 5^1 \implies T(n) \in \Theta(nlog(n))$$

- $T(n) = 5T(\frac{n}{2}) + 2n$

  Tenemos $g(n) = 2n \in O(n)$, por lo tanto

$$\begin{array}{l} a = 5 \\ b = 2 \\ k = 1 \end{array} \implies 5 > 2^1 \implies T(n) \in \Theta(n^{log_2(5)})$$

- $T(n) = \dfrac{\sum_{i=1}^{n}(T(\frac{n}{2}) + i)}{n}$

  Manipulando esta expresión un poco, tenemos

$$\begin{aligned} T(n) &= \frac{\displaystyle\sum_{i=1}^{n}(T(\frac{n}{2}) + i)}{n} \\ &= \frac{\displaystyle\sum_{i=1}^{n} T(\frac{n}{2}) + \sum_{i=1}^{n} i}{n} \\ &= \frac{nT(\frac{n}{2}) + \frac{n(n+1)}{2}}{n} \\ &= \frac{\cancel{n}T(\frac{n}{2})}{\cancel{n}} + \frac{\frac{\cancel{n}(n+1)}{2}}{\cancel{n}} \\ &= T(\frac{n}{2}) + \frac{n+1}{2} \end{aligned}$$

  Por lo que, $g(n) = \frac{(n+1)}{2} \in O(n)$, por lo tanto

$$\begin{array}{l} a = 1 \\ b = 2 \\ k = 1 \end{array} \implies 1 < 2^1 \implies T(n) \in \Theta(n)$$

Pregunta 2

La recurrencia de Perrin está definida como:

$$P(n) = \begin{cases} 3 & \text{si } n = 0 \\ 0 & \text{si } n = 1 \\ 2 & \text{si } n = 2 \\ P(n-2) + P(n-3) & \text{si } 3 \leq n \end{cases}$$

Justificación

Usaremos el principio de "divide y vencerás" para calcular la n-ésima potencia de una matriz de transición en tiempo logarítmico.

Dado que la recurrencia se define por los tres términos anteriores ($P(n-2)$ y $P(n-3)$), se necesita una matriz de transición de dimensión 3×3.

Definimos el vector de estado S_n para el paso n:

$$S_n = \begin{pmatrix} P(n) \\ P(n-1) \\ P(n-2) \end{pmatrix}$$

Para encontrar S_n a partir de S_{n-1}, necesitamos una matriz de transición M tal que $S_n = M \cdot S_{n-1}$.

Las relaciones que definen M son: 1. $P(n) = 0 \cdot P(n-1) + 1 \cdot P(n-2) + 1 \cdot P(n-3)$ 2. $P(n-1) = 1 \cdot P(n-1) + 0 \cdot P(n-2) + 0 \cdot P(n-3)$ 3. $P(n-2) = 0 \cdot P(n-1) + 1 \cdot P(n-2) + 0 \cdot P(n-3)$

La matriz de transición M es:

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Podemos relacionar el estado S_n con el estado base S_2:

$$S_n = M^{n-2} \cdot S_2 \quad \text{para } n \geq 2$$

Donde el vector base es:

$$S_2 = \begin{pmatrix} P(2) \\ P(1) \\ P(0) \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 3 \end{pmatrix}$$

El tiempo de ejecución está dominado por la exponenciación de la matriz M^{n-2}. Dado que la multiplicación de dos matrices 3×3 toma tiempo $O(3^3) = O(1)$ (asumiendo que las operaciones aritméticas elementales son $O(1)$, como se establece en el problema), y la exponenciación se realiza mediante la técnica de divide y vencerás, el tiempo total es $\Theta(\log n)$.

Implementación en C++

A continuación se presenta el código completo en C++ que implementa la lógica anterior.

Utilizamos `long long` para manejar los valores de $P(n)$, ya que el problema no especifica un límite superior para n, y los números de Perrin crecen exponencialmente.

El archivo funcional se encuentra en Perrin.cpp

```
typedef long long ll;
typedef vector<vector<ll>> Matrix;

// Tamaño de la matriz para la recurrencia de Perrin (3x3)
const int K = 3;

Matrix multiply(const Matrix& A, const Matrix& B) {
    Matrix C(K, vector<ll>(K, 0));
    for (int i = 0; i < K; ++i) {
        for (int j = 0; j < K; ++j) {
            for (int k = 0; k < K; ++k) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return C;
}

Matrix power(Matrix M, ll n) {
    // Matriz identidad para inicializar el resultado
    Matrix R(K, vector<ll>(K, 0));
    for (int i = 0; i < K; ++i) R[i][i] = 1;

    while (n > 0) {
        if (n & 1) R = multiply(R, M); // Si n es impar
        M = multiply(M, M);
        n >>= 1; // n = n / 2
    }
    return R;
}

ll perrin(ll n) {
    // Casos base (n=0, 1, 2)
    if (n == 0) return 3;
    if (n == 1) return 0;
    if (n == 2) return 2;
```

```
    // 1. Definir la matriz de transición M
    Matrix M = {
        {0, 1, 1},
        {1, 0, 0},
        {0, 1, 0}
    };

    // 2. Calcular M^(n-2)
    Matrix M_power = power(M, n - 2);

    // 3. Vector base S_2 = {P(2), P(1), P(0)} = {2, 0, 3}
    vector<ll> S_base = {2, 0, 3};

    // 4. Calcular S_n = M_power * S_2. P(n) es el primer elemento.
    // P(n) = M_power*P(2) + M_power*P(1) + M_power*P(0)
    ll P_n = 0;
    for (int i = 0; i < K; ++i) {
        P_n += M_power[0][i] * S_base[i];
    }

    return P_n;
}
```

Pregunta 3