Projeto e Análise de Algoritmos
Exercícios: **Análise Assintótica, Programação Dinâmica e Memoização**

1. **Seja** $P : N \to N$ **uma função definida da seguinte forma:** $P(0) = P(1) = P(2) = P(3) = P(4) = 0$ **e, para** $n \geq 5$**,**

$$P(n) = P(\lfloor \frac{n}{2} \rfloor) + P(\lfloor \frac{n}{2} \rfloor + 1) + P(\lfloor \frac{n}{2} \rfloor + 2) + n.$$

   (a) Escreva um algoritmo recursivo puro que recebe um número n como entrada e retorna o valor exato de $P(n)$. Calcule a complexidade do seu algoritmo.

Algoritmo 1: P(n)

início
    if $n \leq 4$ **then**
        retorne 0;
    else
        retorne $P(\lfloor \frac{n}{2} \rfloor) + P(\lfloor \frac{n}{2} \rfloor + 1) + P(\lfloor \frac{n}{2} \rfloor + 2) + n$;
    end
fim

Complexidadde:

A recurssão gera a seguinte equação de recorrência:

$$T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lfloor \frac{n}{2} \rfloor + 1) + T(\lfloor \frac{n}{2} \rfloor + 2) + n$$

Desprezando as constantes, temos:

$$T(n) = 3T(\frac{n}{2}) + \Theta(1)$$

A partir do Teorema Mestre ($T(n) = aT(\frac{n}{b}) + c$), podemos tirar:

$$a = 3; b = 2; c = 0$$

Logo, a complexidade é:

$$\Theta(n^{\log_2 3})$$

(b) Escreva um algoritmo de programação dinâmica para o mesmo problema e calcule sua complexidade.

i. Primeiro precisamos verificar se uma parte da solução ótima é solução ótima para uma parte do problema:
OK.

ii. Em seguida, criar a recursão:

$$P(n) = \begin{cases} 0, & \text{se } k < 5 \\ P(\lfloor \frac{n}{2} \rfloor) + P(\lfloor \frac{n}{2} \rfloor + 1) + P(\lfloor \frac{n}{2} \rfloor + 2) + n, & \text{se } n \geq 5 \end{cases}$$

iii. E então, criar o algoritmo:

Algoritmo 2: Dinamico(n)

```
1 início
2     criar vetor P[n];
3     P[0] ← P[1] ← P[2] ← P[3] ← P[4] ← 0;
4     i ← 5;
5     repita
6         P[i] ← P[⌊i/2⌋] + P[⌊i/2⌋ + 1] + P[⌊i/2⌋ + 2] + i;
7     até i = n;
8 fim
```

Complexidade: Por conta do *for* (linhas 5 à 7), a complexidade do algoritmo é $\Theta(n)$.

(c) Escreva um algoritmo de memoização para o mesmo problema e calcule sua complexidade.

Algoritmo 3: Memo(n)

início
    criar vetor $P[n]$;
    $i \leftarrow n$;
    $P[n] \leftarrow -1$;
    while $\frac{i}{2} > 0$ **do**
        $P[\lfloor \frac{i}{2} \rfloor] \leftarrow +P[\lfloor \frac{i}{2} \rfloor+1]+P[\lfloor \frac{i}{2} \rfloor+2]+P[\lfloor \frac{i}{2} \rfloor+3]+P[\lfloor \frac{i}{2} \rfloor+4] \leftarrow -1;$
        $i \leftarrow \frac{i}{2}$;
    end
    retorne $MemoRec(P, n)$;
fim

Complexidade: Como o iterador do *while* é sempre dividido pela metade, ele se torna menor ainda a cada nível, assim, a complexidade por ser dada por $\Theta(\log_2 n)$.

Algoritmo 4: MemoRec(P,n)

início
    if $P[n] \neq -1$ **then**
        retorne $P[n]$;
    else
        $P[n] \leftarrow$ $MemoRec(\lfloor \frac{n}{2} \rfloor)+MemoRec(\lfloor \frac{n}{2} \rfloor+1)+MemoRec(\lfloor \frac{n}{2} \rfloor+2)+n;$
        retorne $P[n]$;
    end
fim

Complexidade: Como o n é sempre dividido pela metade a cada recursão, a complexidade por ser dada por $\Theta(\log_2 n)$.

Complexidade Final: Portanto, a complexidade geral é $\Theta(\log n)$

$T_k(n) = n^2$ modos distintos de subir a escada de n degraus com passos de tamanho 1 até k.

Para k¡n, se for dado um passo de tamanho 1, teremos que calcular o número de modos distintos para subir os n-1 degraus restantes. Se for dado um passo de tamanho 2, teremos que calcular n número de modos distintos para subir os n-2 degraus restantes. E assim por diante, até k. Assim, temos:

$$T_k(n) = T_k(n-1) + T_k(n-2) + ... + T_k(n-k)$$

Para k=n, segue-se a mesma lógica, porém, até n-1 e soma-se 1 para contabilizar a possibilidade de dar um passo de tamanho k. Logo:

$$T_k(n) = T_k(n-1) + T_k(n-2) + ... + T_k(1) + 1$$

Quando houver apenas um degrau, haverá apenas uma possibilidade:

$$T_k(1) = 1$$

Caso ocorra $k \geq n$, considera-se $k = n$.

Assim, temos a seguinte equação de recorrência:

$$T_k(n) = \begin{cases} \sum_{i=1}^{k} \mathrm{T}_k(n-i), & \text{se } k < n \\ (\sum_{i=1}^{k-1} \mathrm{T}_k(n-i))+1, & \text{se k} \geq n \\ 1 & \text{se n=1} \end{cases}$$

2. **Prove as seguintes afirmações sobre notação assintótica:**

- $n^3/100 - 25n^2 - 100n + 7$ é $\Omega(\mathrm{n}^2)$ e $\Theta(\mathrm{n}^3)$
  - $^3/100 - 25n^2 - 100n + 7 \geq \Omega(\mathrm{n}^2)$
    $n^3/100 - 25n^2 - 100n^2 \geq \Omega(\mathrm{n}^2)$
    $n^3/100 - 125n^2 \geq \Omega(\mathrm{n}^2)$
    $n^3/100 - 12500n^2/100 \geq \Omega(\mathrm{n}^2)$
    $2n^3/200 - 25000n^2/200 \geq \Omega(\mathrm{n}^2)$
    $2n^3/200 - (n * n^2)/200 \geq \Omega(\mathrm{n}^2)$
    $2n^3/200 - (n^3)/200 \geq \Omega(\mathrm{n}^2)$
    $n^3/200 \geq \Omega(\mathrm{n}^2)$
    Assim, temos: $n_0' \geq 25000$ e $c_1 = 1/200$
  - $n^3/100 - 25n^2 - 100n + 7 \leq \Theta(\mathrm{n}^3)$
    $n^3/100 \leq \mathrm{O}(\mathrm{n}^3)$
    Assim, temos: $n_0'' = 1$ e $c_2 = 1/100$
  - Portanto, para ser válido, devemos ter:
    $n_0 \geq 25000$
    $c_1 = 1/200$
    $c_2 = 1/100$
- $77n^3 - 13n^2 + 29n - 5$ é $O(n^4)$ e $\Omega(\mathrm{n}^3)$
  - $77n^3 - 13n^2 + 29n - 5 \leq \mathrm{O}(\mathrm{n}^4)$
    $77n^3 + 29n^3 \leq \mathrm{O}(\mathrm{n}^4)$
    $106n^3 \leq \mathrm{O}(\mathrm{n}^4)$
    $n * n^3 \leq \mathrm{O}(\mathrm{n}^4)$
    $n^4 \leq \mathrm{O}(\mathrm{n}^4)$
    Assim, temos: $n_0' \geq 106$ e $c_1 = 1$
  - $77n^3 - 13n^2 + 29n - 5 \geq \mathrm{O}(\mathrm{n}^3)$
    $77n^3 - 13n^2 - 5n^2 \geq \mathrm{O}(\mathrm{n}^3)$
    $77n^3 - 18n^2 \geq \mathrm{O}(\mathrm{n}^3)$
    $77n^3 - n * n^2 \geq \mathrm{O}(\mathrm{n}^3)$
    $77n^3 - n^3 \geq \mathrm{O}(\mathrm{n}^3)$
    $76n^3 \geq \mathrm{O}(\mathrm{n}^3)$
    Assim, temos: $n_0'' \geq 18$ e $c_2 = 76$
  - Portanto, para ser válido, devemos ter:
    $n_0 \geq 106$
    $c_1 = 1$
    $c_2 = 76$

- $34n \log_7 n^2 + 13n$ é $\Omega(n)$ e $O(n^2)$
  - $34n \log_7 n^2 + 13n \geq \Omega(n)$
    $13n \geq O(n^2)$
    Assim, temos: $n'_0 = 1$ e $c_1 = 13$
  - $34n \log_7 n^2 + 13n \leq O(n^2)$
    $34n * 2 \log_7 n + 13n \leq O(n^2)$
    $34n * 2 \log_7 7 + 13n \leq O(n^2)$
    $34n * 2 * 1 + 13n \leq O(n^2)$
    $34n * 2 + 13n \leq O(n^2)$
    $68n \leq O(n^2)$
    $81n \leq O(n^2)$
    $81n^2 \leq O(n^2)$
    Assim, temos $n''_0 = 7$ e $c_2 = 81$
  - Portanto, devemos ter:
    $n_0 = 7$
    $c_1 = 13$
    $c_2 = 81$