Resolvendo

Lista 1

Universidade Federal de Minas Gerais
Departamento de Computação
Projeto e Análise de Algoritmos - 2024.2
Professor: Marcio Costa Santos
Lista 1

Exercício 1. Determine a função de complexidade do algoritmo abaixo e indique sua complexidade de melhor caso, caso médio e pior caso

- ***Pseudocódigo Q1***
  - **Entrada**: Vetor de n inteiros a
  - $cnt \leftarrow 0;$
  - **para todo** $i \leftarrow 0$ até $n - 1$ **faça**
    - **se** $a[i]\%2 = 0$ **então**
      - $cnt \leftarrow cnt + 1;$
  - **retorna** cnt;

Contando a quantidade de passos:

- ***Pseudocódigo Q1***
  - **Entrada**: Vetor de n inteiros a [0]
  - $cnt \leftarrow 0;$ [C_1]
  - **para todo** $i \leftarrow 0$ até $n - 1$ **faça** [Inicial: C_2; por ciclo: C_3]
    - **se** $a[i]\%2 = 0$ **então** [Por ciclo: C_4]
      - $cnt \leftarrow cnt + 1;$ [Caso verdadeiro: C_5]
  - **retorna** cnt; [0]

Somando os passos:

- $T(n) = C_1 + C_2 + \sum_{i=0}^{n-1} (C_3 + C_4 + P_5 * C_5)$
- $T(n) = C_{1,2} + \sum_{i=0}^{n-1} (C_{3,4} + P_5 * C_5)$
- $T(n) = C_{1,2} + n * C_{3,4} + n * P_5 * C_5$
- $O(T(n)) = O(C_{1,2}) + O(n * C_{3,4}) + O(n * P_5 * C_5)$
- $O(T(n)) = O(1) + O(n) + O(n)$
- $O(T(n)) = O(n)$

Onde temos que:

- $C_1 = 1$ (1 atribuição)
- $C_2 = 2$ (1 atribuição; 1 comparação inicial)
- $C_3 = 2$ (1 comparação; 1 incremento)
- $C_4 = 3$ (1 índice; 1 divisão; 1 comparação)x
- $C_5 = 2$ (1 soma; 1 atribuição)
- $P_5 = 0,5 \text{ (chance de ser par) ou } \frac{\text{número de pares em } a}{n}$
- $C_{1,2} = C_1 + C_2$
- $C_{3,4} = C_3 + C_4$

Temos então que sua função de complexidade é:

- $T(n) = C_{1,2} + n * C_{3,4} + n * P_5 * C_5$

1.a. Melhor caso

Como o algoritmo em questão conta a quantidade de números pares, sua execução variará de acordo com a quantidade de números pares. Sendo assim, o melhor caso é quando não há números pares no vetor a.

Dessa forma, ele realizará todos os passos do algoritmo, porém não entrará no bloco condicional, ou seja, $n * P_5 * C_5 = 0$.

- $MelhorT(n) = C_{1,2} + n * C_{3,4} + n * P_5 * C_5$
- $MelhorT(n) = C_{1,2} + n * C_{3,4}$
- $\Omega(T(n)) = \Omega(C_{1,2}) + \Omega(n * C_{3,4})$
- $\Omega(T(n)) = \Omega(1) + \Omega(n)$
- $\Omega(T(n)) = \Omega(n)$

1.b. Caso médio

Em média, teremos que a quantidade de números pares será exatamente a metade dos números do vetor a, ou seja, $P_5 = 0,5$. Dessa forma, temos que:

- $MédioT(n) = C_{1,2} + n * C_{3,4} + n * P_5 * C_5$
- $MédioT(n) = C_{1,2} + n * C_{3,4} + n * 0,5 * C_5$
- $\Theta(T(n)) = \Theta(C_{1,2}) + \Theta(n * C_{3,4}) + \Theta(n * 0,5 * C_5)$
- $\Theta(T(n)) = \Theta(1) + \Theta(n) + \Theta(n)$
- $\Theta(T(n)) = \Theta(n)$

1.c. Pior caso

O pior caso é quando todos os números do vetor a são pares, ou seja, $P_5 = 1$. Dessa forma, temos que:

- $PiorT(n) = C_{1,2} + n * C_{3,4} + n * P_5 * C_5$
- $PiorT(n) = C_{1,2} + n * C_{3,4} + n * 1 * C_5$
- $PiorT(n) = C_{1,2} + n * C_{3,4} + n * C_5$
- $O(T(n)) = O(C_{1,2}) + O(n * C_{3,4}) + O(n * C_5)$
- $O(T(n)) = O(1) + O(n) + O(n)$
- $O(T(n)) = O(n)$

Exercício 2. Determine a função de complexidade do algoritmo abaixo e indique sua complexidade de melhor caso, caso médio e pior caso

- ***Pseudocódigo Q2***
  - **Entrada:** Matrizes $n \times n$ A e B
  - C $\leftarrow$ matriz vazia;
  - **para todo** $i \leftarrow 0$ até $n - 1$ **faça**
    - **para todo** $j \leftarrow 0$ até $n - 1$ **faça**
      - $C[i, j] \leftarrow 0$;
      - **para todo** $k \leftarrow 0$ até $n - 1$ **faça**
        - $C[i, j] \leftarrow C[i, j] + A[i, k] * B[k, j]$;
  - **retorna** C;

Contando a quantidade de passos:

- ***Pseudocódigo Q2***
  - **Entrada:** Matrizes $n \times n$ A e B
  - C $\leftarrow$ matriz vazia; [C_1]
  - **para todo** $i \leftarrow 0$ até $n - 1$ **faça** [Inicial: C_2; por ciclo: C_3]
    - **para todo** $j \leftarrow 0$ até $n - 1$ **faça** [Inicial: C_4; por ciclo: C_5]
      - $C[i, j] \leftarrow 0$; [C_6]
      - **para todo** $k \leftarrow 0$ até $n - 1$ **faça** [Inicial: C_7; por ciclo: C_8]
        - $C[i, j] \leftarrow C[i, j] + A[i, k] * B[k, j]$; [$C_9$]
  - **retorna** C;

Onde temos que:

- $C_1 = n^2$ (Preenchimento de matriz com zeros)
- $C_2 = 2$ (1 atribuição; 1 comparação inicial)
- $C_3 = 2$ (1 comparação; 1 incremento)
- $C_4 = 2$ (1 atribuição; 1 comparação inicial)
- $C_5 = 2$ (1 comparação; 1 incremento)
- $C_6 = 3$ (2 acesso ao índice; 1 atribuição)
- $C_7 = 2$ (1 atribuição; 1 comparação inicial)
- $C_8 = 2$ (1 comparação; 1 incremento)
- $C_9 = 5$ (4*(2 acessos ao índice); 1 atribuição; 2 operações matemáticas)

Somando os passos:

- $T(n) = C_1 + C_2 + \sum_{i=0}^{n-1} \left(C_3 + C_4 + \sum_{j=0}^{n-1} \left(C_5 + C_6 + C_7 + \sum_{k=0}^{n-1} (C_8 + C_9) \right) \right)$

Simplificando os limites do somatório pela quantidade de elementos:

- $T(n) = C_1 + C_2 + \sum_{i=1}^{n} \left(C_3 + C_4 + \sum_{j=1}^{n} (C_5 + C_6 + C_7 + \sum_{k=1}^{n} (C_8 + C_9)) \right)$
- $T(n) = C_{1,2} + \sum_{i=1}^{n} \left(C_{3,4} + \sum_{j=1}^{n} (C_{5,6,7} + \sum_{k=1}^{n} (C_{8,9})) \right)$
- $T(n) = C_{1,2} + \sum_{i=1}^{n} \left(C_{3,4} + \sum_{j=1}^{n} (C_{5,6,7} + n * C_{8,9}) \right)$
- $T(n) = C_{1,2} + \sum_{i=1}^{n} (C_{3,4} + n * C_{5,6,7} + n * n * C_{8,9})$
- $T(n) = C_{1,2} + n * C_{3,4} + n * n * C_{5,6,7} + n * n * n * C_{8,9}$
- $T(n) = C_{1,2} + n * C_{3,4} + n^2 * C_{5,6,7} + n^3 * C_{8,9}$

Temos então que sua função de complexidade é:

- $T(n) = C_{1,2} + n * C_{3,4} + n^2 * C_{5,6,7} + n^3 * C_{8,9}$

E podemos considerar que é dominada assintoticamente por n^3.

- $O(T(n)) = O(C_{1,2}) + O(n * C_{3,4}) + O(n^2 * C_{5,6,7}) + O(n^3 * C_{8,9})$
- $O(T(n)) = O(1) + O(n) + O(n^2) + O(n^3)$
- $O(T(n)) = O(n^3)$

2.a. Melhor caso

O melhor caso ocorre quando todos os elementos das matrizes são zero, pois a multiplicação de qualquer número por zero resultará em zero, então as somas podem ser feitas sem custo adicional (não há operação significativa de multiplicação ou soma). No entanto, como o número de operações ainda é $O(n^3)$ (devido aos três loops aninhados), a complexidade do melhor caso ainda é $O(n^3)$.

2.b. Caso médio

No caso médio, onde as matrizes contêm valores variados, a complexidade também será dominada pelas operações de multiplicação e soma nos três loops aninhados. Portanto, a complexidade do caso médio será também $O(n^3)$.

2.c. Pior caso

O **algoritmo 2** é um algoritmo de multiplicação de matrizes, onde a matriz resultante é preenchida com zeros e depois é feita a multiplicação de cada elemento da matriz resultante com os elementos das matrizes A e B.

De modo geral, o custo computacional desse algoritmo poderá variar de acordo com duas situações:

- A ordem da matriz;
- O tamanho dos valores das matrizes.

É esperado que uma multiplicação entre números grandes tenha um custo computacional maior do que uma multiplicação entre números pequenos ou que sejam iguais a zero.

Ao desconsiderarmos essas duas situações, teremos que todos os três casos (melhor, médio e pior) terão o mesmo custo computacional, ou seja, $O(n^3)$. Isso porque em todos os casos, o algoritmo terá que percorrer todos os elementos das matrizes A e B e realizar a multiplicação de cada elemento da matriz resultante.

Exercício 3. Considere o seguinte algoritmo

- ***Pseudocódigo Q3***
  - **Entrada:** vetor de inteiros A, tamanho n de A
  - **para todo** $i \leftarrow 2$ até n **faça**
    - $chave \leftarrow A[i]$;
    - $j \leftarrow i - 1$;
    - **enquanto** $j > 0$ e $A[j] > chave$ **faça**
      - $A[j+1] \leftarrow A[j]$;
      - $j \leftarrow j - 1$;
    - $A[j+1] \leftarrow chave$;
  - **retorna** A;

3.a. Simule a execução do algoritmo para o vetor [3, 5, 2, 8, 9]

- Variáveis:
  - $n = 5; A = [3, 5, 2, 8, 9]; i = 2; chave(A[i]) = 5; j = 1; A[j] = 3;$
  - $n = 5; A = [3, 5, 2, 8, 9]; i = 3; chave(A[i]) = 2; j = 2; A[j] = 5;$
    - $A = [3, 5, 5, 8, 9];$
    - $A = [3, 2, 5, 8, 9];$

- $n = 5; A = [3, 2, 5, 8, 9]; i = 3; chave(A[i]) = 2; j = 1; A[j] = 3;$
  - $A = [3, 3, 5, 8, 9];$
  - $A = [2, 3, 5, 8, 9];$
- $n = 5; A = [2, 3, 5, 8, 9]; i = 4; chave(A[i]) = 8; j = 3; A[j] = 5;$
- $n = 5; A = [2, 3, 5, 8, 9]; i = 5; chave(A[i]) = 9; j = 4; A[j] = 2;$
- FIM: $A = [2, 3, 5, 8, 9];$

3.b. O que esse algoritmo faz?

Esse é o algoritmo conhecido como Insertion Sort que ordena uma lista de números de forma crescente. Seu algoritmo consiste em percorrer toda a lista de números, sempre checando se o número atual é maior que todos os anteriores, assim colocando o novo número na posição correta para que a lista fique ordenada.

3.c. Qual sua complexidade de pior caso?

Calculando o custo de cada passo:

- ***Pseudocódigo Q3***
  - **Entrada:** vetor de inteiros A, tamanho n de A
  - **para todo** $i \leftarrow 2$ até n **faça** $[\text{inicial: } C_1, \text{por ciclo: } C_2]$
    - $chave \leftarrow A[i]; [C_3]$
    - $j \leftarrow i - 1; [C_4]$
    - **enquanto** $j > 0$ e $A[j] > chave$ **faça** $[\text{inicial: } C_5, \text{por ciclo: } C_6]$
      - $A[j+1] \leftarrow A[j]; [C_7]$
      - $j \leftarrow j - 1; [C_8]$
    - $A[j+1] \leftarrow chave; [C_9]$
  - **retorna** A;

Gerando a equação:

- $T(n) = C_1 + \sum_{i=2}^{n}(C_2 + C_3 + C_4 + C_5 + C_9) + \sum_{j=i-1}^{j \geq 0 \wedge}(C_6 + C_7 + C_8)$
- $T(n) = C_1 + \sum_{i=2}^{n}\left(C_{2,3,4,5,9} + \sum_{j=1}^{i-1} C_{6,7,8}\right)$
- $T(n) = C_1 + (n-1) \cdot \left(C_{2,3,4,5,9} + \sum_{j=1}^{i-1} C_{6,7,8}\right)$
- $T(n) = C_1 + n \cdot C_{2,3,4,5,9} - C_{2,3,4,5,9} + (n-1) \cdot \left(\sum_{j=1}^{i-1} C_{6,7,8}\right)$

[JV: Fiquei confuso quanto ao que fazer no caso do somatório que aumenta de acordo com a variação do i]

Descrevendo as constantes:

- $C_1 = 2$ (1 atribuição; 1 comparação inicial)
- $C_2 = 2$ (1 comparação; 1 incremento)
- $C_3 = 2$ (1 atribuição; 1 acesso ao índice)
- $C_4 = 2$ (1 atribuição; 1 operação matemática)
- $C_5 = 3$ (2 comparações; 1 acesso ao índice)
- $C_6 = 3$ (2 comparações; 1 acesso ao índice)
- $C_7 = 3$ (2 acessos ao índice; 1 operação; 1 atribuição)
- $C_8 = 2$ (1 atribuição; 1 operação matemática)
- $C_9 = 3$ (1 operação matemática; 1 acesso ao índice; 1 atribuição)

Para esse algoritmo, o pior caso ocorre quando a comparação $A[j] > chave$ sempre for verdadeira. Esse caso se dá quando a lista está ordenada de forma decrescente, pois sua chave sempre será menor que todos os itens percorridos pela variável j, assim fazendo a maior quantidade de trocas possíveis.

[JV: Sinto que houve um salto lógico aqui, não entendi ao certo como justificar alcançar o n^2]

Dessa forma, temos que a complexidade de pior caso é $O(n^2)$.

3.d. Qual sua complexidade de melhor caso?

O melhor caso ocorre quando a lista já está ordenada de forma crescente, pois a chave sempre será maior que o primeiro item verificado pela variável j, assim poupando a verificação com os demais.

[JV: eu precisaria explicar mais sobre a decomposição das constantes em $O()$?]

Nesse caso, o algoritmo percorrerá todos os elementos da lista, porém não realizará nenhuma troca, sendo assim, a complexidade de melhor caso é $O(n)$.

Exercício 4. Considere o seguinte algoritmo

- ***Pseudocódigo Q4***
  - **Entrada:** vetor de inteiros A, tamanho n de A

- **para todo** $i \leftarrow 1$ até $n-1$ **faça**
  - **para todo** $j \leftarrow n$ até $i+1$ **faça**
    - **se** $A[j] < A[j-1]$ **então**
      - troque $A[j]$ com $A[j-1]$;
- **retorna** A;

4.a. Simule a execução do algoritmo para o vetor [3, 5, 2, 8, 9]

- Variáveis: $A = [3, 5, 2, 8, 9]; n = 5; i = 1; j = 5; A[j] = 9; A[j-1] = 8;$
- Variáveis: $A = [3, 5, 2, 8, 9]; n = 5; i = 1; j = 4; A[j] = 8; A[j-1] = 2;$
- Variáveis: $A = [3, 5, 2, 8, 9]; n = 5; i = 1; j = 3; A[j] = 2; A[j-1] = 5;$ Troca
- Variáveis: $A = [3, 2, 5, 8, 9]; n = 5; i = 1; j = 2; A[j] = 2; A[j-1] = 3;$ Troca
- Variáveis: $A = [2, 3, 5, 8, 9]; n = 5; i = 2; j = 5; A[j] = 9; A[j-1] = 8;$
- Variáveis: $A = [2, 3, 5, 8, 9]; n = 5; i = 2; j = 4; A[j] = 8; A[j-1] = 5;$
- Variáveis: $A = [2, 3, 5, 8, 9]; n = 5; i = 2; j = 3; A[j] = 5; A[j-1] = 3;$
- Variáveis: $A = [2, 3, 5, 8, 9]; n = 5; i = 3; j = 5; A[j] = 9; A[j-1] = 8;$
- Variáveis: $A = [2, 3, 5, 8, 9]; n = 5; i = 3; j = 4; A[j] = 8; A[j-1] = 5;$
- Variáveis: $A = [2, 3, 5, 8, 9]; n = 5; i = 4; j = 5; A[j] = 9; A[j-1] = 8;$
- Fim

4.b. O que esse algoritmo faz?

Este é o algoritmo de ordenação conhecido como Bubble Sort. Ele percorre toda a lista de números, começando à esquerda e colocando todos os menores números à esquerda e os maiores à direita, ou seja, ordenando de forma crescente. Ele faz isso percorrendo a lista e sempre que o número na posição verificada for menor que o anterior, ele troca. Em cada uma das iterações de i ele encontrará o i-ésimo menor número e o colocará na posição i do vetor.

4.c. Qual sua complexidade de pior caso?

O pior caso é quando a lista está ordenada de forma decrescente, precisando então realizar a maior quantidade de trocas possíveis.

Como são dois loops, um dentro do outro, ambos percorrendo aproximadamente n elementos, temos que a complexidade de pior caso é $O(n^2)$.

[JV: preciso depois descobrir qual é a função $f(n)$?]

[JV: Talvez usaria aquela ideia de $\frac{n*(n-1)}{2}$, mas que igualmente seria $O(n^2)$]

4.d. Qual sua complexidade de melhor caso?

O seu melhor caso ocorre quando a lista já está ordenada de forma crescente.

Mesmo que o algoritmo não precise fazer troca alguma, ainda assim ele percorre aproximadamente n^2 elementos, sendo assim, a complexidade de melhor caso segue sendo $O(n^2)$.

Exercício 5. Determine um limite superior assintótico para as funções abaixo (de preferência o mais apertado possível)

Para essa questão é importante considerarmos que:

- O : Limite Superior
- o : Limite Superior estrito
- Θ : Equivalência
- ω : Limite Inferior estrito
- Ω : Limite Inferior
- $O(n!) > O(2^n) > O(n^2) > O(n \log n) > O(n) > O(\log n) > O(1)$

> Limite Superior (O)
>
> $f = O(g)$ Existem n_0 e c tal que: $f(n) \leq c \cdot g(n)$ para todo $n \geq n_0$

- $f = o(g); f(n) < c * g(n); n \geq n_0$

5.1. $2n^3 + n^4 - 1$

- $f(n) = 2n^3 + n^4 - 1$
- $O(f(n)) = O(2n^3 + n^4 - 1)$
- $O(f(n)) = O(n^4)$

5.2. $2^n + 5\log n + n^2$

- $f(n) = 2^n + 5\log n + n^2$
- $O(f(n)) = O(2^n + 5\log n + n^2)$
- $O(f(n)) = O(2^n)$

5.3. $\log_{10} n + \log_3 10$

- $f(n) = \log_{10} n + \log_3 10$
- $O(f(n)) = O(\log_{10} n + \log_3 10)$
- $O(f(n)) = O(\log_{10} n)$
- $O(f(n)) = O(\log n)$

5.4. $n + n\log n + \log n$

- $f(n) = n + n\log n + \log n$
- $O(f(n)) = O(n + n\log n + \log n)$
- $O(f(n)) = O(n\log n)$

5.5. $4^n + 2^n + n$

- $f(n) = 4^n + 2^n + n$
- $O(f(n)) = O(4^n + 2^n + n)$
- $O(f(n)) = O(4^n)$
- $O(f(n)) = O(2^{2+n})$
- $O(f(n)) = O(2^n)$

Tabela resumindo as respostas das questões 5 e 6

# Equação	Função	Limite Superior (O) (Q 5, 6)
.1	$2n^3 + n^4 - 1$	n^4
.2	$2^n + 5\log n + n^2$	2^n
.3	$\log_{10} n + \log_3 10$	$\log n$
.4	$n + n\log n + \log n$	$n\log n$
.5	$4^n + 2^n + n$	2^n

Exercício 6. Determine um limite superior assintótico para as funções abaixo (de preferência o mais apertado possível) - [IGNORADA POR SER EXATAMENTE IGUAL AO EXERCÍCIO 5]

- 6.1. $2n^3 + n^4 - 1$
- 6.2. $2^n + 5\log n + n^2$
- 6.3. $\log_{10} n + \log_3 10$
- 6.4. $n + n\log n + \log n$
- 6.5. $4^n + 2^n + n$

Exercício 7. Determine um limite superior assintótico restrito para as funções abaixo (de preferência o mais apertado possível)

Para essa questão é importante considerarmos que:

- O : Limite Superior
- o : Limite Superior estrito
- Θ : Equivalência
- ω : Limite Inferior estrito
- Ω : Limite Inferior
- $O(n!) > O(2^n) > O(n^2) > O(n\log n) > O(n) > O(\log n) > O(1)$

Limite Superior Estrito (o)

f = o(g) para todo $c > 0$ existe n_0 tal que: $f(n) < c * g(n)$ para todo $n \geq n_0$

- $f = o(g); f(n) < c * g(n); n \geq n_0$

7.1. $2n^3 + n^4 - 1$

- $f(n) = 2n^3 + n^4 - 1$
- $o(f(n)) = o(2n^3 + n^4 - 1)$
- $o(f(n)) = o(n^4)$

Achando uma função $g(n)$ que seja maior que $f(n)$:

- $g(n) = n^5$

Se considerarmos que o $n_0 = 1$ e $c = 1$, temos então que:

- $g(n) > c * f(n)$
- $n^5 > 2n^3 + n^4 - 1$

Sabemos que para todos os valores de $n \geq 1$ a função $f(n)$ é menor que $g(n)$, sendo assim, $n^5 = o(f(n))$.

7.2. $2^n + 5 \log n + n^2$

- $f(n) = 2^n + 5 \log n + n^2$
- $o(f(n)) = o(2^n + 5 \log n + n^2)$
- $o(f(n)) = o(2^n)$

Achando uma função $g(n)$ que seja maior que $f(n)$:

- $g(n) = 3^n$

Se considerarmos que o $n_0 = 1$ e $c = 1$, sabemos que para todos os valores de $n \geq 1$ a função $f(n)$ é menor que $g(n)$, sendo assim, $3^n = o(f(n))$.

7.3. $\log_{10} n + \log_3 10$

- $f(n) = \log_{10} n + \log_3 10$
- $o(f(n)) = o(\log_{10} n + \log_3 10)$
- $o(f(n)) = o(\log_{10} n) + C_1$
- $o(f(n)) = o(\log n)$

Achando uma função $g(n)$ que seja maior que $f(n)$:

- $g(n) = n$

Se considerarmos que o $n_0 = 10$ e $c = 1$, sabemos que para todos os valores de $n \geq 10$ a função $f(n)$ é menor que $g(n)$, sendo assim, $n = o(f(n))$.

7.4. $n + n \log n + \log n$

- $f(n) = n + n \log n + \log n$
- $o(f(n)) = o(n + n \log n + \log n)$
- $o(f(n)) = o(n \log n)$

Achando uma função $g(n)$ que seja maior que $f(n)$:

- $g(n) = n^2$

Se considerarmos que o $n_0 = 10$ e $c = 1$, sabemos que para todos os valores de $n \geq 2$ a função $f(n)$ é menor que $g(n)$, sendo assim, $n^2 = o(f(n))$.

7.5. $4^n + 2^n + n$

- $f(n) = 4^n + 2^n + n$
- $o(f(n)) = o(4^n + 2^n + n)$

- $o(f(n)) = o(4^n)$

Achando uma função $g(n)$ que seja maior que $f(n)$:

- $g(n) = 5^n$

Se considerarmos que o $n_0 = 1$ e $c = 1$, sabemos que para todos os valores de $n \geq 1$ a função $f(n)$ é menor que $g(n)$, sendo assim, $5^n = o(f(n))$.

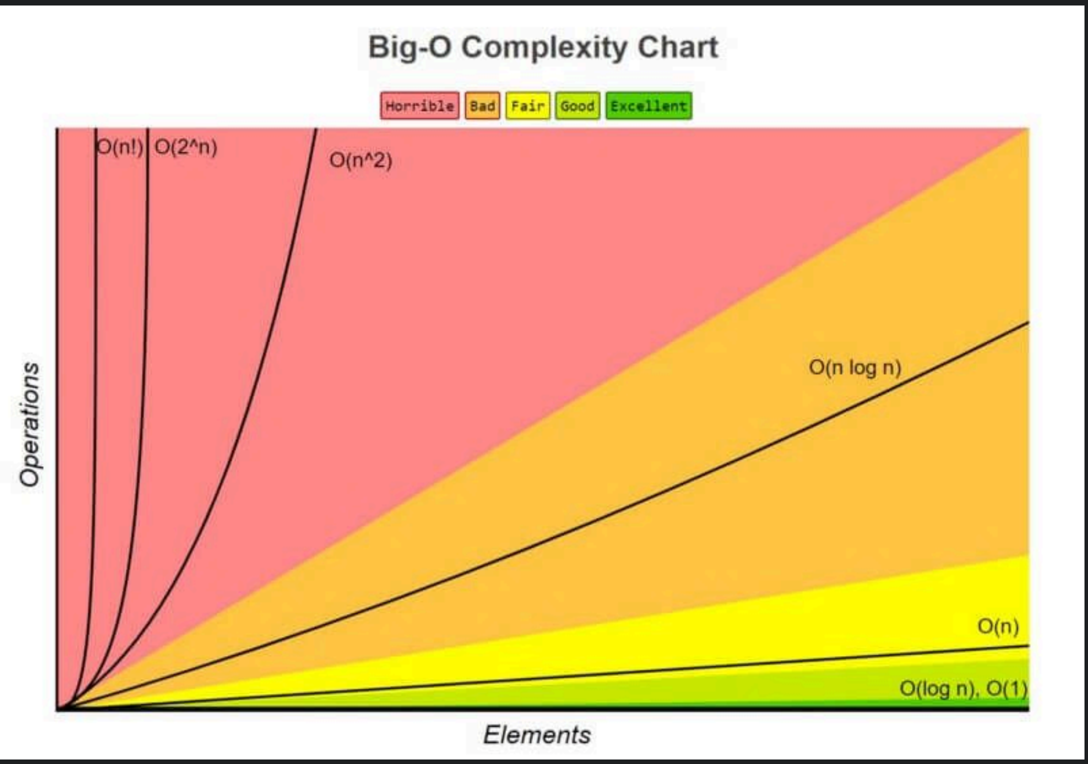
Exercício 8. Determine um limite inferior assintótico para as funções abaixo (de preferência o mais apertado possível)

Para essa questão é importante considerarmos que:

- O : Limite Superior
- o : Limite Superior estrito
- Θ : Equivalência
- ω : Limite Inferior estrito
- Ω : Limite Inferior
- $O(n!) > O(2^n) > O(n^2) > O(n \log n) > O(n) > O(\log n) > O(1)$

> Limite Inferior (Ω)
>
> $f = \Omega(g)$ Existem n_0 e c tal que: $f(n) \geq c * g(n)$ para todo $n \geq n_0$

- $f = \Omega(g); f(n) \geq c * g(n); n \geq n_0$



8.1. $2n^3 + n^4 - 1$

- $f(n) = 2n^3 + n^4 - 1$
- $\Omega(f(n)) = \Omega(2n^3 + n^4 - 1)$
- $\Omega(f(n)) = \Omega(n^4)$

Achando uma função $g(n)$ que seja maior ou igual que $f(n)$:

- $g(n) = n^4$

Se considerarmos que o $n_0 = 1$ e $c = 1$, sabemos que para todos os valores de $n \geq n_0$ a função $f(n)$ é maior ou igual que $g(n)$, sendo assim, $5^n = \Omega(f(n))$.

8.2. $2^n + 5 \log n + n^2$

- $f(n) = 2^n + 5 \log n + n^2$
- $\Omega(f(n)) = \Omega(2^n + 5 \log n + n^2)$
- $\Omega(f(n)) = \Omega(2^n)$

Achando uma função $g(n)$ que seja maior ou igual que $f(n)$:

- $g(n) = 2^n$

Se considerarmos que o $n_0 = 1$ e $c = 1$, sabemos que para todos os valores de $n \geq n_0$ a função $f(n)$ é maior ou igual que $g(n)$, sendo assim, $2^n = \Omega(f(n))$.

8.3. $\log_{10} n + \log_3 10$

- $f(n) = \log_{10} n + \log_3 10$
- $\Omega(f(n)) = \Omega(\log_{10} n + \log_3 10)$
- $\Omega(f(n)) = \Omega(\log_{10} n + C_1)$
- $\Omega(f(n)) = \Omega(\log n)$

Achando uma função $g(n)$ que seja maior ou igual que $f(n)$:

- $g(n) = \log n$

Se considerarmos que o $n_0 = 10$ e $c = 1$, sabemos que para todos os valores de $n \geq n_0$ a função $f(n)$ é maior ou igual que $g(n)$, sendo assim, $\log n = \Omega(f(n))$.

8.4. $n + n \log n + \log n$

- $f(n) = n + n \log n + \log n$
- $\Omega(f(n)) = \Omega(n + n \log n + \log n)$
- $\Omega(f(n)) = \Omega(n \log n)$

Achando uma função $g(n)$ que seja maior ou igual que $f(n)$:

- $g(n) = n \log n$

Se considerarmos que o $n_0 = 10$ e $c = 1$, sabemos que para todos os valores de $n \geq n_0$ a função $f(n)$ é maior ou igual que $g(n)$, sendo assim, $n \log n = \Omega(f(n))$.

8.5. $4^n + 2^n + n$

- $f(n) = 4^n + 2^n + n$
- $\Omega(f(n)) = \Omega(4^n + 2^n + n)$
- $\Omega(f(n)) = \Omega(4^n)$

Achando uma função $g(n)$ que seja maior ou igual que $f(n)$:

- $g(n) = 4^n$

Se considerarmos que o $n_0 = 1$ e $c = 1$, sabemos que para todos os valores de $n \geq n_0$ a função $f(n)$ é maior ou igual que $g(n)$, sendo assim, $4^n = \Omega(f(n))$.

Exercício 9. Determine um limite inferior assintótico restrito para as funções abaixo (de preferência o mais apertado possível)

Para essa questão é importante considerarmos que:

- $O(n!) > O(2^n) > O(n^2) > O(n \log n) > O(n) > O(\log n) > O(1)$

> Limite Inferior Assintótico Estrito (ω)
>
> $f = \omega(g)$ Para todo $c > 0$ existe n_0 tal que: $f(n) > c * g(n)$ para todo $n \geq n_0$

- $f = \omega(g) \forall c > 0 \exists n_0 | f(n) > c * g(n) \forall n \geq n_0$

9.1. $2n^3 + n^4 - 1$ || $\omega(n^3)$ com $c = 4$ e $n_0 = 10$

- **Limite inferior restrito:** $\omega(n^3)$.
- **Constantes:** $c = 4$, $n_0 = 10$.

- **Cálculo para $n = 10$:**

$$f(10) = 2 \cdot 10^3 + 10^4 - 1 = 2000 + 10000 - 1 = 11999,$$

$$c \cdot g(10) = 4 \cdot 10^3 = 4000.$$

Resultado: $11999 > 4000$.

Resposta: $\boxed{\omega(n^3)}$ com $c = 4$ e $n_0 = 10$.

9.2. $2^n + 5\log n + n^2$ || $\omega(2^{n/2})$ com $c = 2$ e $n_0 = 10$

- **Limite inferior restrito:** $\omega(2^{n/2})$.
- **Constantes:** $c = 2$, $n_0 = 10$.
- **Cálculo para $n = 10$:**

$$f(10) = 2^{10} + 5\log 10 + 10^2 = 1024 + 5 \cdot 1 + 100 = 1129,$$

$$c \cdot g(10) = 2 \cdot 2^{10/2} = 2 \cdot 32 = 64.$$

Resultado: $1129 > 64$.

Resposta: $\boxed{\omega(2^{n/2})}$ com $c = 2$ e $n_0 = 10$.

9.3. $\log_{10} n + \log_3 10$ || $\omega(1)$ com $c = 3$ e $n_0 = 10$

- **Limite inferior restrito:** $\omega(1)$.
- **Constantes:** $c = 3$, $n_0 = 10$.
- **Cálculo para $n = 10$:**

$$f(10) = \log_{10} 10 + \log_3 10 = 1 + 2.095 \approx 3.095,$$

$$c \cdot g(10) = 3 \cdot 1 = 3.$$

Resultado: $3.095 > 3$.

Resposta: $\boxed{\omega(1)}$ com $c = 3$ e $n_0 = 10$.

9.4. $n + n\log n + \log n$ || $\omega(n)$ com $c = 2$ e $n_0 = 100$

- **Limite inferior restrito:** $\omega(n)$.
- **Constantes:** $c = 2$, $n_0 = 100$.
- **Cálculo para $n = 100$:**

$$f(100) = 100 + 100\log 100 + \log 100 = 100 + 200 + 2 = 302,$$

$$c \cdot g(100) = 2 \cdot 100 = 200.$$

Resultado: $302 > 200$.

Resposta: $\boxed{\omega(n)}$ com $c = 2$ e $n_0 = 100$.

9.5. $4^n + 2^n + n$ || $\omega(2^n)$ com $c = 2$ e $n_0 = 10$

- **Limite inferior restrito:** $\omega(2^n)$.
- **Constantes:** $c = 2$, $n_0 = 10$.
- **Cálculo para $n = 10$:**
  - $f(10) = 4^{10} + 2^{10} + 10 = 1048576 + 1024 + 10 = 1049610$
  - $c \cdot g(10) = 2 \cdot 2^{10} = 2 \cdot 1024 = 2048$

    Resultado: $1049610 > 2048$.

Resposta: $\boxed{\omega(2^n)}$ com $c = 2$ e $n_0 = 10$.

Exercício 10. Determine uma equivalência assintótica para as funções abaixo

> $f = \Theta(g)$
>
> Existem n_0, c_1 e c_2 tal que:
>
> - $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ para todo $n \geq n_0$

10.1. $2n^3 + n^4 - 1$ || R: $\Theta(n^4)$

- **Equivalência:** $\Theta(n^4)$.
- **Constantes:**
  - $c_1 = 1$, $c_2 = 3$, $n_0 = 2$.
- **Cálculo para $n = 2$:**
  - $c_1 \cdot n^4 = 1 \cdot 2^4 = 16$,
  - $f(2) = 2 \cdot 2^3 + 2^4 - 1 = 16 + 16 - 1 = 31$,
  - $c_2 \cdot n^4 = 3 \cdot 2^4 = 48$.

    Desigualdade: $16 \leq 31 \leq 48 \checkmark$.

Resposta: $\boxed{\Theta(n^4)}$ com $c_1 = 1$, $c_2 = 3$, $n_0 = 2$.

10.2. $2^n + 5\log n + n^2$ || R: $\Theta(2^n)$

- **Equivalência:** $\Theta(2^n)$.
- **Constantes:**
  - $c_1 = 1$, $c_2 = 2$, $n_0 = 5$.
- **Cálculo para $n = 5$:**
  - $c_1 \cdot 2^n = 1 \cdot 2^5 = 32$,
  - $f(5) = 2^5 + 5\log 5 + 5^2 = 32 + 5 \cdot 2.321 + 25 \approx 32 + 11.605 + 25 = 68.605$,
  - $c_2 \cdot 2^n = 2 \cdot 2^5 = 64$.

    Ajuste: Para $n \geq 6$, $f(n) \leq 2 \cdot 2^n$.

Resposta: $\boxed{\Theta(2^n)}$ com $c_1 = 1$, $c_2 = 2$, $n_0 = 5$.

10.3. $\log_{10} n + \log_3 10$ || R: $\Theta(\log n)$

- **Equivalência:** $\Theta(\log n)$.
- **Constantes:**
  - $c_1 = 1$, $c_2 = 3$, $n_0 = 100$.
- **Cálculo para $n = 100$:**
  - $c_1 \cdot \log n = \log_{10} 100 = 2$,
  - $f(100) = \log_{10} 100 + \log_3 10 = 2 + 2.095 \approx 4.095$,
  - $c_2 \cdot \log n = 3 \cdot \log_{10} 100 = 6$.

    Desigualdade: $2 \leq 4.095 \leq 6 \checkmark$.

Resposta: $\boxed{\Theta(\log n)}$ com $c_1 = 1$, $c_2 = 3$, $n_0 = 100$.

10.4. $n + n\log n + \log n$ || R: $\Theta(n\log n)$

- **Equivalência:** $\Theta(n\log n)$.
- **Constantes:**
  - $c_1 = 1, c_2 = 4, n_0 = 2$.
- **Cálculo para $n = 2$:**
  - $c_1 \cdot n\log n = 2\log 2 \approx 1.386$,
  - $f(2) = 2 + 2\log 2 + \log 2 \approx 2 + 1.386 + 0.693 \approx 4.079$,
  - $c_2 \cdot n\log n = 4 \cdot 2\log 2 \approx 5.545$.
    Desigualdade: $1.386 \leq 4.079 \leq 5.545$ ✓.

Resposta: $\boxed{\Theta(n\log n)}$ com $c_1 = 1$, $c_2 = 4$, $n_0 = 2$.

10.5. $4^n + 2^n + n$ || R: $\Theta(4^n)$

- **Equivalência:** $\Theta(4^n)$.
- **Constantes:**
  - $c_1 = 1, c_2 = 2, n_0 = 1$.
- **Cálculo para $n = 1$:**
  - $c_1 \cdot 4^n = 4$,
  - $f(1) = 4 + 2 + 1 = 7$,
  - $c_2 \cdot 4^n = 2 \cdot 4 = 8$.
    Desigualdade: $4 \leq 7 \leq 8$ ✓.

Resposta: $\boxed{\Theta(4^n)}$ com $c_1 = 1$, $c_2 = 2$, $n_0 = 1$.

Tabela resumindo as respostas das questões 5 a 10

# Equação	Função	Limite Superior Estrito (o) (Q 7)	Limite Superior (O) (Q 5, 6)	Equivalência (Θ) (Q 10)	Limite Inferior (Ω) (Q 8)	Limite Inferior Estrito (ω) (Q 9)
.1	$2n^3 + n^4 - 1$	n^5	n^4	n^4	n^4	n^5
.2	$2^n + 5\log n + n^2$	3^n	2^n	2^n	2^n	3^n
.3	$\log_{10} n + \log_3 10$	n	$\log n$	$\log n$	$\log n$	n
.4	$n + n\log n + \log n$	n^2	$n\log n$	$n\log n$	$n\log n$	n^2
.5	$4^n + 2^n + n$	5^n	2^n	4^n	4^n	5^n

Exercício 11. Dadas funções $f(n)$, $h(n)$ e $g(n)$ prove que

11.1. Se $f(n) = O(g(n))$ e $g(n) = O(h(n))$ então $f(n) = O(h(n))$

- **Prova:**
  - Por definição:
    a. $f(n) = O(g(n))$ implica que existem $c_1 > 0$ e $n_1 \geq 0$ tais que $f(n) \leq c_1 \cdot g(n)$ para $n \geq n_1$.
    b. $g(n) = O(h(n))$ implica que existem $c_2 > 0$ e $n_2 \geq 0$ tais que $g(n) \leq c_2 \cdot h(n)$ para $n \geq n_2$.
  - Escolha $n_0 = \max(n_1, n_2)$. Para $n \geq n_0$:
    - $f(n) \leq c_1 \cdot g(n) \leq c_1 \cdot (c_2 \cdot h(n)) = (c_1 c_2) \cdot h(n)$
  - Portanto, $f(n) = O(h(n))$ com $c = c_1 c_2$ e $n_0 = \max(n_1, n_2)$.

11.2. $f(n) = O(f(n))$

- **Prova:**
  - Por definição, $f(n) = O(f(n))$ se existem $c > 0$ e $n_0 \geq 0$ tais que $f(n) \leq c \cdot f(n)$ para $n \geq n_0$.
  - Escolha $c = 1$ e $n_0 = 0$. Então, para todo $n \geq 0$:
    - $f(n) \leq 1 \cdot f(n)$
  - Portanto, $f(n) = O(f(n))$.

11.3. Se $f(n) = \Omega(g(n))$ e $g(n) = \Omega(h(n))$ então $f(n) = \Omega(h(n))$

- **Prova:**
  - Por definição:
    a. $f(n) = \Omega(g(n))$ implica que existem $c_1 > 0$ e $n_1 \geq 0$ tais que $f(n) \geq c_1 \cdot g(n)$ para $n \geq n_1$.
    b. $g(n) = \Omega(h(n))$ implica que existem $c_2 > 0$ e $n_2 \geq 0$ tais que $g(n) \geq c_2 \cdot h(n)$ para $n \geq n_2$.
  - Escolha $n_0 = \max(n_1, n_2)$. Para $n \geq n_0$:
    - $f(n) \geq c_1 \cdot g(n) \geq c_1 \cdot (c_2 \cdot h(n)) = (c_1 c_2) \cdot h(n)$
  - Portanto, $f(n) = \Omega(h(n))$ com $c = c_1 c_2$ e $n_0 = \max(n_1, n_2)$.

11.4. $f(n) = \Omega(f(n))$

- **Prova:**
  - Por definição, $f(n) = \Omega(f(n))$ se existem $c > 0$ e $n_0 \geq 0$ tais que $f(n) \geq c \cdot f(n)$ para $n \geq n_0$.
  - Escolha $c = 1$ e $n_0 = 0$. Então, para todo $n \geq 0$:
    - $f(n) \geq 1 \cdot f(n)$
  - Portanto, $f(n) = \Omega(f(n))$.

11.5. $f(n) \neq o(f(n))$

- **Prova:**
  - Por definição, $f(n) = o(f(n))$ se, para **todo** $c > 0$, existe $n_0 \geq 0$ tal que $f(n) < c \cdot f(n)$ para $n \geq n_0$.
  - Dividindo ambos os lados por $f(n)$ (assumindo $f(n) > 0$):
    - $1 < c$
  - Essa desigualdade **não é válida** para $c = 1$, pois $1 \not< 1$.
  - Portanto, $f(n) \neq o(f(n))$.

11.6. $f(n) \neq \omega(f(n))$

- **Prova:**
  - Por definição, $f(n) = \omega(f(n))$ se, para **todo** $c > 0$, existe $n_0 \geq 0$ tal que $f(n) > c \cdot f(n)$ para $n \geq n_0$.
  - Dividindo ambos os lados por $f(n)$ (assumindo $f(n) > 0$):
    - $1 > c$
  - Essa desigualdade **não é válida** para $c = 1$, pois $1 \not> 1$.
  - Portanto, $f(n) \neq \omega(f(n))$.

Exercício 12. Prove que $n^3 \neq O(n^2)$

Definição de O: $f(n) = O(g(n))$ se existem $c > 0$ e $n_0 \geq 0$ tais que $f(n) \leq c \cdot g(n)$ para $n \geq n_0$.

- **Hipótese para contradição:** Suponha que $n^3 = O(n^2)$.
- **Desigualdade derivada:** Então, existem $c > 0$ e n_0 tais que:
  - $n^3 \leq c \cdot n^2 \text{ para todo } n \geq n_0.$
- **Simplificação:** Divida ambos os lados por n^2 (para $n > 0$):
  - $n \leq c$
- **Contradição:** A desigualdade $n \leq c$ não pode ser verdadeira para **todo** $n \geq n_0$, pois n cresce indefinidamente. Enquanto que c é constante.

Conclusão: Não existem c e n_0 que satisfaçam a definição. Portanto: $\boxed{n^3 \neq O(n^2)}$.

Exercício 13. Prove que $n \neq O(\log n)$

Definição de O: $f(n) = O(g(n))$ se existem $c > 0$ e $n_0 \geq 0$ tais que $f(n) \leq c \cdot g(n)$ para $n \geq n_0$.

- **Hipótese para contradição:** Suponha que $n = O(\log n)$.
- **Desigualdade derivada:** Então, existem $c > 0$ e n_0 tais que:
  - $n \leq c \cdot \log n \text{ para todo } n \geq n_0$
- **Análise de crescimento:**
  - n cresce **linearmente**, enquanto $\log n$ cresce **logaritmicamente**.
  - Para $n \to \infty$, $\log n$ é insignificante comparado a n.
- **Limite assintótico:** Calcule o limite: $\lim_{n\to\infty} \frac{n}{\log n} = \infty$
  - Isso mostra que n ultrapassa $c \cdot \log n$ para qualquer $c > 0$.
- **Contradição:** Para n suficientemente grande, $n > c \cdot \log n$, violando a suposição.

Conclusão: Não existem c e n_0 que satisfaçam a definição. Portanto: $\boxed{n \neq O(\log n)}$.

Exercício 14. Prove que $\sum_{i=1}^{n} i = \Theta(n^2)$, utilizando uma prova por indução

Para provar por indução, primeiro precisamos provar o $\sum_{i=1}^{n} i = O(n^2)$ e depois o $\sum_{i=1}^{n} i = \Omega(n^2)$.

Para provar que $\sum_{i=1}^{n} i = \Omega(n^2)$, pela definição do Ω, temos que:

- $f = \Omega(g)$ Existem n_0 e c tal que: $f(n) \geq c \cdot g(n)$ para todo $n \geq n_0$
- Caso base: $n_0 = 1$ e $c = \frac{1}{10}$
  - $\sum_{i=1}^{1} i \geq \frac{1}{10} \cdot 1^2$
  - $1 \geq \frac{1}{10}$
- Hipótese de indução: $n = k$
  - $\sum_{i=1}^{k} i \geq \frac{1}{10} \cdot k^2$
- Passo indutivo: $n = k + 1$
  - $\sum_{i=1}^{k+1} i \geq \frac{1}{10} \cdot (k+1)^2$
- Resolvendo então indutivamente, partindo da hipótese de indução e alcançando o passo indutivo:
  - $\sum_{i=1}^{k} i \geq \frac{1}{10} \cdot k^2$
  - $\sum_{i=1}^{k} i + (k+1) \geq \frac{1}{10} \cdot k^2 + (k+1)$
    - Provando que $\frac{1}{10} \cdot k^2 + (k+1) \geq \frac{1}{10} \cdot (k+1)^2$:
      - $\frac{1}{10} \cdot k^2 + (k+1) \geq \frac{1}{10} \cdot k^2 + \frac{1}{10} \cdot (2k+1)$
      - $k + 1 \geq \frac{2k+1}{10}$
      - $k + 1 \geq \frac{k}{5} + \frac{1}{10}$
  - $\sum_{i=1}^{k+1} i \geq \frac{1}{10} \cdot k^2 + (k+1) \geq \frac{1}{10} \cdot (k+1)^2$
  - $\sum_{i=1}^{k+1} i \geq \frac{1}{10} \cdot (k+1)^2$

Exercício 15. Prove que $\sum_{i=1}^{n} \frac{1}{k} = \Theta(\log n)$

Definição de Θ

Precisamos encontrar $c_1 > 0$, $c_2 > 0$ e $n_0 \geq 1$ tais que:

- $c_1 \cdot \log n \leq H(n) \leq c_2 \cdot \log n \text{ para todo } n \geq n_0\$$

- **Passo 1: Limite Inferior ($H(n) \geq c_1 \cdot \log n$)**
  - **Estratégia:** Agrupar os termos em blocos que dobram de tamanho e mostrar que cada bloco contribui com pelo menos $\frac{1}{2}$.
    - **Agrupamento:**
      - Para $n \geq 1$, escreva n como $2^m - 1$ (onde $m = \lfloor \log_2(n+1) \rfloor$).
      - Divida a soma em blocos:
      - $\underbrace{1}_{\text{Bloco 0}} + \underbrace{\frac{1}{2}}_{\text{Bloco 1}} + \underbrace{\frac{1}{3} + \frac{1}{4}}_{\text{Bloco 2}} + \underbrace{\frac{1}{5} + \cdots + \frac{1}{8}}_{\text{Bloco 3}} + \cdots$
  - **Limite por bloco:**
    - Cada bloco i contém 2^i termos, todos menores ou iguais a $\frac{1}{2^i}$. Por exemplo:
      - Bloco 0: $1 \geq \frac{1}{2}$.
      - Bloco 1: $\frac{1}{2} \geq \frac{1}{2}$.
      - Bloco 2: $\frac{1}{3} + \frac{1}{4} \geq \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$.
      - Bloco i: $\sum_{k=2^i}^{2^{i+1}-1} \frac{1}{k} \geq 2^i \cdot \frac{1}{2^{i+1}} = \frac{1}{2}$.
  - **Total de blocos:**
    - Se $n \geq 2^m - 1$, existem m blocos completos. Como $m \geq \log_2(n+1) - 1$, temos:

        - $H(n) \geq \frac{1}{2} \cdot m \geq \frac{1}{2} \cdot (\log_2 n - 1)$.
    - Convertendo para logaritmo natural ($\log_2 n = \frac{\ln n}{\ln 2}$):
        - $H(n) \geq \frac{1}{2\ln 2} \cdot \ln n - \frac{1}{2}$.
    - Para $n \geq 4$, $\frac{1}{2\ln 2} \cdot \ln n - \frac{1}{2} \geq \frac{1}{4} \cdot \ln n$.
  - **Escolha:** $c_1 = \frac{1}{4}; n_0 = 4$.
- **Passo 2: Limite Superior ($H(n) \leq c_2 \cdot \log n$)**
  - **Estratégia:** Comparar a soma com uma série telescópica.
    - **Desigualdade telescópica:**
      - Observe que para $k \geq 1$:
      - $\frac{1}{k} \leq \int_{k-1}^{k} \frac{1}{x} dx$ (opcional, mas evitamos integrais na prova final).
      - **Alternativa sem integrais:**
        - Agrupe os termos de forma que cada bloco tenha soma ≤ 1.
    - **Agrupamento alternativo:**
      - Para $n \geq 1$, divida a soma em blocos de tamanho 2^i:
      - $\underbrace{1}_{\text{Bloco 0}} + \underbrace{\frac{1}{2} + \frac{1}{3}}_{\text{Bloco 1}} + \underbrace{\frac{1}{4} + \cdots + \frac{1}{7}}_{\text{Bloco 2}} + \cdots$
    - **Limite por bloco:**
      - Cada bloco i contém 2^i termos, todos maiores ou iguais a $\frac{1}{2^{i+1}}$. Por exemplo:
        - Bloco 0: $1 \leq 1$.
        - Bloco 1: $\frac{1}{2} + \frac{1}{3} \leq 1$.
        - Bloco 2: $\frac{1}{4} + \cdots + \frac{1}{7} \leq 4 \cdot \frac{1}{4} = 1$.
    - **Total de blocos:**
      - Se $n \leq 2^m - 1$, existem m blocos. Portanto:
        - $H(n) \leq 1 + 1 + 1 + \cdots + 1 = m \leq \log_2 n + 1$.
      - Convertendo para logaritmo natural:
        - $H(n) \leq \frac{\ln n}{\ln 2} + 1$.
      - Para $n \geq 2$, $\frac{\ln n}{\ln 2} + 1 \leq 2 \cdot \ln n$.
  - **Escolha:** $c_2 = 2; n_0 = 2$.
- **Conclusão**
  - Existem constantes $c_1 = \frac{1}{4}$, $c_2 = 2$ e $n_0 = 4$ tais que:
  - $\frac{1}{4} \cdot \ln n \leq H(n) \leq 2 \cdot \ln n \text{ para todo } n \geq 4$.
  - Portanto: $\boxed{\sum_{k=1}^{n} \frac{1}{k} = \Theta(\log n)}$.