PUC Minas

Lista 1 – Função de complexidade

INFORMAÇÕES DOCENTE						
CURSO: ENGENHARIA DE SOFTWARE	**DISCIPLINA:** FUNDAMENTOS DE PROJETO E ANÁLISE DE ALGORITMOS	**TURNO**	MANHÃ	TARDE	NOITE	**PERÍODO/SALA:** 5º
					x	
PROFESSOR (A): João Paulo Carneiro Aramuni						

Lista 1 - Gabarito

Função de complexidade

1) O algoritmo abaixo implementa uma função que encontra o maior valor dentro de uma lista.

```
1  def max(lista):
2      temp = lista[0]  # Inicializa temp com o primeiro elemento da lista
3      for i in range(1, len(lista)):  # Percorre os elementos do índice 1 até o final
4          if temp < lista[i]:  # Se encontrar um elemento maior que temp
5              temp = lista[i]  # Atualiza temp com esse valor
6      return temp  # Retorna o maior valor encontrado
```

a) Quais são as operações mais relevantes?
As operações mais significativas são aquelas que dependem de n, pois dominam o crescimento da função conforme n aumenta:
As operações que mais impactam o crescimento do tempo de execução são:
1. Comparações dentro do if → (n - 1) vezes
2. Atribuições dentro do if (no pior caso) → (n - 1) vezes
3. Iterações do for → (n - 1) vezes

Essas operações contribuem para a complexidade assintótica.

b) Quanto tempo se gasta em relação ao tamanho n da lista? (Qual a função de complexidade?)

```
1  def max(lista):
2      temp = lista[0]  # (1) Atribuição inicial
3      for i in range(1, len(lista)):  # (n - 1) iterações
4          if temp < lista[i]:  # (n - 1) comparações
5              temp = lista[i]  # No pior caso, (n - 1) atribuições
6      return temp  # (1) Retorno do valor máximo
```

Contagem de operações:
1. Inicialização: temp = lista[0] → 1 operação.
2. Laço for: O loop roda de i = 1 até i = n - 1, ou seja, (n - 1) iterações.
3. Comparação dentro do if: Acontece em (n - 1) vezes.
4. Atribuição dentro do if (pior caso - lista em ordem crescente): Pode ocorrer no máximo (n - 1) vezes.
5. Retorno do valor máximo: 1 operação.

Soma das operações:
Total de operações:
$1 + (n-1) + (n-1) + (n-1) + 1 = 3(n-1) + 2$
Ou seja: $f(n) = 3n - 1$

2) O algoritmo abaixo retorna o maior e o menor valor dentro de uma lista de números.

```
def max_min(lista):
    temp = [lista[0], lista[0]]  # Inicializa temp com o primeiro elemento como maior e menor
    for i in range(1, len(lista)):  # Percorre a lista do índice 1 até o final
        if temp[0] < lista[i]:  # Se encontrar um número maior que temp[0]
            temp[0] = lista[i]  # Atualiza temp[0] com esse valor (maior número)
        elif temp[1] > lista[i]:  # Se encontrar um número menor que temp[1]
            temp[1] = lista[i]  # Atualiza temp[1] com esse valor (menor número)
    return temp  # Retorna uma lista [maior_valor, menor_valor]
```

a) Quais são as operações mais relevantes?

As operações mais significativas são aquelas que dependem de n, pois dominam o crescimento da função conforme n aumenta:
As operações que mais impactam o crescimento do tempo de execução são:
1. Comparações no if e elif
2. Atribuições dentro do if e elif
3. Execução do loop for

Essas operações contribuem para a complexidade assintótica.

b) Quanto tempo se gasta em relação ao tamanho n da lista? (Qual a função de complexidade?)

```
def max_min(lista):
    temp = [lista[0], lista[0]]  # (1) Inicializa temp com o primeiro elemento como maior e menor
    for i in range(1, len(lista)):  # (n - 1) iterações do loop
        if temp[0] < lista[i]:  # (n - 1) comparações no pior caso
            temp[0] = lista[i]  # No pior caso, (n - 1) atribuições
        elif temp[1] > lista[i]:  # (n - 1) comparações no pior caso
            temp[1] = lista[i]  # No pior caso, (n - 1) atribuições
    return temp  # (1) Retorno da lista [maior_valor, menor_valor]
```

Contagem de operações:

1. Inicialização do vetor temp → 1 operação.
2. Loop for → Executado (n - 1) vezes.
3. Comparação no if → No pior caso, ocorre em (n - 1) iterações.
4. Atribuição no if (atualização do maior valor) → Ocorre no máximo (n - 1) vezes.
5. Comparação no elif → Ocorre no máximo (n - 1) vezes.
6. Atribuição no elif (atualização do menor valor) → Ocorre no máximo (n - 1) vezes.
7. Retorno de temp → 1 operação.

Soma das operações:

Total de operações:

$1 + (n - 1) + (n - 1) + (n - 1) + (n - 1) + (n - 1) + 1 = 5(n - 1) + 2$

Ou seja: $f(n) = 5n - 3$
