PUC Minas

Lista 2 – Complexidade Ciclomática

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 2 - Gabarito

Complexidade Ciclomática

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
```

1. Monte o grafo de fluxo de controle da função:
- Identifique os nós (representando os pontos de decisão e instruções da função).
- Identifique as arestas (representando as transições entre os nós).

2. Calcule a complexidade ciclomática da função usando a fórmula:

$$M = E - N + 2P$$

- Onde: E é o número de arestas no grafo.
- N é o número de nós no grafo.
- P é o número de componentes conexos (neste caso, $P = 1$, pois a função é uma unidade única).

3. Interprete o valor da complexidade ciclomática:
- Explique o que significa o valor obtido para o número de caminhos independentes no código.

4. Descreva os caminhos independentes possíveis no grafo de fluxo de controle para essa função.

Cálculo: Função max(lista)

I. Representação da função em fluxo de controle
Passos do fluxo de controle:
1. Início da função.
2. Inicialização da variável temp.
3. Início do laço for.
4. Verificação da condição if temp < lista[i].
- Se verdadeiro: Executa temp = lista[i].
- Se falso: Passa direto para a próxima iteração.
5. Retorno da variável temp.

Observação sobre o passo 2: Vamos supor que a inicialização da variável temp fosse temp = [] ao invés de temp = lista[0]:
- Neste caso, não seria contado como um nó porque: A inicialização de temp não afetaria nenhuma decisão no código. O laço for e o if não dependeriam do valor inicial de temp. O fluxo de controle não mudaria por causa dessa atribuição.

II. Estruturando o Grafo de fluxo
Um grafo de controle representa os caminhos possíveis da execução:
- Nó: Representa um ponto de decisão ou instrução;
- Aresta: Representa a transição entre nós.
- Componentes conexos (P): A função é uma unidade única, então $P = 1$ (é um bloco único de código sem chamadas externas a outras funções ou estruturas separadas).

Nós (N):
1. Início da função.
2. Inicialização de temp.
3. Início do laço for.
4. Verificação do if.
5. Ação dentro do if (temp = lista[i]).
6. Próxima iteração ou saída do laço.
7. Retorno final.
Número total de nós: $N = 7$.

Arestas (E):
1. Do Início para a inicialização de temp: 1 aresta.
2. Da inicialização de temp para o laço for: 1 aresta.
3. Do laço for para o if: 1 aresta.
4. Condição verdadeira no if para temp = lista[i]: 1 aresta.
5. Condição falsa no if para a próxima iteração do laço: 1 aresta.
6. De temp = lista[i] para a próxima iteração do laço: 1 aresta.
7. Saída do laço para o retorno: 1 aresta.
Número total de arestas: $E = 7$.

III. Aplicando a fórmula

Agora, usamos a fórmula da complexidade ciclomática:
$M = E - N + 2P$
Substituímos os valores:
$M = 7 - 7 + 2(1) \rightarrow M = 2$

IV. Interpretando o resultado

- A complexidade ciclomática da função é 2.
- Isso significa que há 2 caminhos independentes no grafo de fluxo de controle:
    - O caminho onde temp nunca é atualizado (ou seja, a condição if temp < lista[i] nunca é verdadeira).
    - O caminho onde temp é atualizado pelo menos uma vez (o bloco do if é executado).

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

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

1. Monte o grafo de fluxo de controle da função:
- Identifique os nós (representando os pontos de decisão e instruções da função).
- Identifique as arestas (representando as transições entre os nós).

2. Calcule a complexidade ciclomática da função usando a fórmula:

$$M = E - N + 2P$$

- Onde: E é o número de arestas no grafo.
- N é o número de nós no grafo.
- P é o número de componentes conexos (neste caso, $P = 1$, pois a função é uma unidade única).

3. Interprete o valor da complexidade ciclomática:
- Explique o que significa o valor obtido para o número de caminhos independentes no código.

4. Descreva os caminhos independentes possíveis no grafo de fluxo de controle para essa função.

Cálculo: Função max_min(lista)

I. Representação da função em fluxo de controle

Passos do fluxo de controle:
1. Início da função.
2. Inicialização da variável temp com [lista[0], lista[0]].
3. Início do laço for.
4. Verificação da condição if temp[0] < lista[i].
- Se verdadeiro: Executa temp[0] = lista[i].
- Se falso: Passa para a verificação do elif.
5. Verificação da condição elif temp[1] > lista[i].
- Se verdadeiro: Executa temp[1] = lista[i].
- Se falso: Passa direto para a próxima iteração.
6. Retorno da variável temp.

II. Estruturando o Grafo de fluxo

Um grafo de controle representa os caminhos possíveis da execução:
- Nó: Representa um ponto de decisão ou instrução;
- Aresta: Representa a transição entre nós.
- Componentes conexos (P): A função é uma unidade única, então $P = 1$ (é um bloco único de código sem chamadas externas a outras funções ou estruturas separadas).

Nós (N):
1. Início da função.
2. Inicialização de temp.
3. Início do laço for.
4. Verificação do if.
5. Ação dentro do if (temp[0] = lista[i]).
6. Verificação do elif.
7. Ação dentro do elif (temp[1] = lista[i]).
8. Próxima iteração ou saída do laço.
9. Retorno final.
Número total de nós: $N = 9$.

Arestas (E):
1. Do Início para a inicialização de temp: 1 aresta.
2. Da inicialização de temp para o laço for: 1 aresta.
3. Do laço for para o if: 1 aresta.

4. Condição verdadeira no if para temp[0] = lista[i]: 1 aresta.
5. De temp[0] = lista[i] para a próxima iteração do laço: 1 aresta.
6. Condição falsa no if para o elif: 1 aresta.
7. Condição verdadeira no elif para temp[1] = lista[i]: 1 aresta.
8. De temp[1] = lista[i] para a próxima iteração do laço: 1 aresta.
9. Condição falsa no elif para a próxima iteração do laço: 1 aresta.
10. Saída do laço para o retorno: 1 aresta.

Número total de arestas: $E = 10$.

III. Aplicando a fórmula

Agora, usamos a fórmula da complexidade ciclomática:

$M = E - N + 2P$

Substituímos os valores:

$M = 10 - 9 + 2(1)$ -> $M = 3$

IV. Interpretando o resultado

- A complexidade ciclomática da função é 3.
- Isso significa que há 3 caminhos independentes no grafo de fluxo de controle:
    - O caminho onde nenhuma das condições (if ou elif) são verdadeiras (nenhum valor de temp é atualizado).
    - O caminho onde apenas o if é verdadeiro (temp[0] é atualizado).
    - O caminho onde o if é falso, mas o elif é verdadeiro (temp[1] é atualizado).