UNIVERSIDADE FEDERAL UBERLÂNDIA
CAMPUS MONTE CARMELO
CURSO BACHAREL EM SISTEMAS DE INFORMAÇÃO

RHUAN FLORES CUNHA FERNANDES
CLÉSIO RODRIGUES DA SILVA JÚNIOR

1° TRABALHO DE ESTRUTURA DE DADOS II

Monte Carmelo-MG
04/2023

SUMÁRIO

1 DEFINIÇÃO DE TAD

O texto deverá ser digitado em Fonte Times New Roman, tamanho 12. O espaçamento entre linhas deverá ser configurado como 1,5. Além disso, as margens deverão obedecer à seguinte configuração: esquerda e superior 3cm; direita e inferior 2cm.

TAD é um modelo matemático de uma estrutura de dados e as operações suportadas nesta estrutura. Pode ser definido como a tupla (v, op) onde v é o conjunto de valores e op o conjunto das operações aplicadas sobre estes valores. Esta tupla é conhecida como assinatura do TAD. Ele é definido em função das operações que suporta e não na maneira como são implementadas. Um TAD pode ser reutilizado e sua manutenção não afeta os programas que os usam, bastando recopilá-los.

2 COMO UTILIZAR O TAD NO PROBLEMA

Para definir um TAD em programação, descrevem-se dois módulos separados: um é a interface de acesso (assinatura do TAD), que informa o nome e os protótipos documentados de todas as operações suportadas, o outro módulo é a implementação da estrutura de dados, assim como de todas as rotinas que implementam suas operações. Permite que o programador use o TAD sem realmente conhecer a estrutura de dados usada ou como as operações foram implementadas.

Em C normalmente se guarda as implementações dos TADs em arquivos a parte, e estes são chamados nos programas que os usam pela diretiva #include. No caso deste trabalho a TAD foi definida com o nome de Complexidade e possuía os atributos de (long int contadorMat, {iniciarCp(Complexidade *c), contarCp(long int *c, int add), finalizarCp(Complexidade *c, int n)}).

O atributo da TAD contadorMat serve para realizar a marcação de quantas vezes houve alguma operação (adição, subtração, multiplicação e/ou divisão) no programa. As funções iniciarCp, contarCp e finalizarCp servem respectivamente para iniciar a TAD Complexidade, atribuindo os valores para os atributos de 0, para incrementar o contador, alegando que houve alguma operação no código e, por fim, para finalizar a TAD e mostrar o resultado de desempenho do programa ou da parte analisada.

Após o desenvolvimento da TAD no arquivo “TAD.h”, ele foi importado nos 3 programas com #define. O valor de análise “n” foi definido como 30, assim todos os “for” do exercício entravam no loop 30 vezes. Assim, a TAD Complexidade analisava o desempenho do programa através de quantas operações os programas faziam. A tabela abaixo mostra essa relação.

Programa	Desempenho
1	Contador = 60 entre O(n * Log(n) e O(n^2)
2	Contador = 1346268 entre O(n^3) e O(2^n)
3	Contador = 54930 entre O(n^3) e O(2^n)

Tablela I: Relação do Programa x Desempenho

Podemos ver que no programa 1, que havia poucos “for”, o programa rodou em excelente qualidade, entretanto os programas 2 e 3 se o “n” for muito grande pode haver alguma demora para rodar o código. Isso acontece por causa da recursividade na questão 2 e por 3 “for” na questão 3, assim, caso seja desejado loops que repetem muitas vezes, é aconselhável mudar a lógica do programa.

3 LIMITAÇÃO DO TAD COMPLEXIDADE NO PROBLEMA

A TAD Complexidade nesse problema, apenas analisa as equações feitas, não analisando o problema como um todo, por exemplo, se houvesse algum acesso a rede no problema e isso retirasse desempenho do programa por demora, ela não consideraria. Possíveis soluções para isso seria adicionar novas funções e atributos para a TAD através da biblioteca “time.h”, adicionando um contador que analisa o tempo e o clock gasto pelo programa. Assim, poderia ter uma analise mais precisa e abrangente sobre os programas.