IGED: Um Ambiente para Ensino de Estruturas de Dados com Análise Automática de Complexidade de Algoritmos

Erick Jonh F. Costa[1], Jorge Gabriel G. de S. Ramos[1], Yuri de A. Malheiros Barbosa[1], Gilberto Farias de S. Filho[2], Alisson V. Brito[2]

[1]Departamento de Ciências Exatas – Universidade Federal da Paraíba(UFPB)
Rio Tinto – PB – Brasil

[2]Centro de Informática – Universidade Federal da Paraíba(UFPB)
João Pessoa – PB – Brasil

{erick.costa, jgabriel, yuri}@dce.ufpb.br, {gilberto, alisson}@ci.ufpb.br

Resumo. *Duas características de um algoritmo têm importância fundamental no seu projeto ou uso: se ele permite chegar à resposta desejada (corretude) e quão eficiente ele é no uso dos recursos computacionais (complexidade). Portanto, é necessário que esses dois aspectos sejam enfatizados durante o ensino de desenvolvimento de algoritmos. Este trabalho propõe uma solução para a medição da eficiência de algoritmos por meio da implementação de um novo componente para a ferramenta IGED (Interpretador Gráfico de Estruturas de Dados), propondo o uso combinado das abordagens empírica e assintótica para avaliação de complexidade dos algoritmos.*

Palavras Chave: *Análise Assintótica; Medição de Complexidade; Interpolação de Lagrange.*

1. Introdução

A análise de algoritmos compreende, em geral, duas dimensões: a corretude e a complexidade. A análise de corretude visa determinar se o algoritmo chega na solução desejada, enquanto que a análise de complexidade determina qual o custo desse algoritmo na busca da solução. A ferramenta IGED – Interpretador Gráfico de Estrutura de Dados (Sousa et al., 2012) – foi projetada para auxiliar no ensino de tais disciplinas; desta forma, é importante que a ferramenta realize automaticamente as análises de corretude e complexidade (eficiência) dos algoritmos implementados pelos alunos. Neste sentido, o presente trabalho propõe um novo componente para o IGED, permitindo ao aluno a determinação empírica da eficiência de seu algoritmo, podendo compará-lo a outras implementações.

2. Trabalhos Relacionados

Segundo Cormen (2002), analisar um algoritmo significa prever os recursos de que ele necessitará. Em geral, memória e processamento são a preocupações primordiais, mas frequentemente é o tempo de computação que se deseja medir. O tempo de execução de um algoritmo será representado por uma função de custo T(n) que representa a medida de tempo necessária para executar um algoritmo para um problema de tamanho n. É importante enfatizar que T(n) não representa diretamente o tempo de execução, mas o número de vezes que as operações relevantes são executadas.

O sistema ACE, Automatic Complexity Evaluator [Le Métayer 1988], é capaz de analisar algoritmos de propósito geral implementados em linguagem funcional. Partindo-se do programa inicial, uma função de complexidade de tempo é derivada. Esta função é transformada em uma função não recursiva equivalente, de acordo com o princípio de indução de recursão, usando uma biblioteca pré-definida de definições recursivas. Quando o sistema realiza a conversão das equações a expressão final de complexidade será uma composição de funções bem conhecidas, tais como funções exponenciais, logarítmicas, etc. O objetivo do ACE é a análise teórica da complexidade dos algoritmos, por isto não é considerado o custo de funções primitivas, desta forma a complexidade assintótica e o número de chamadas recursivas necessárias para a avaliação do programa possuem a mesma ordem de magnitude.

O ANAC [Barbosa et al. 2001] é uma ferramenta para o cálculo da complexidade de algoritmos. Basicamente é um programa de análise-síntese de códigos fontes escritos em uma linguagem procedural pré-estabelecida. Este sistema tem como componentes analisadores léxico e sintático, análogos a um compilador, mas sua síntese não é uma listagem de código ou conjunto de operações, mas sim o desenvolvimento de um cálculo baseado nas estruturas presentes no programa fonte, gerando como resultado uma equação de complexidade. O ANAC possui um tratamento sobre as equações de complexidade geradas com o objetivo de simplificá-las, facilitando a análise do resultado pelo usuário. entretanto o ANAC não avalia a complexidade de algoritmos recursivos, já que os mesmos não avaliam a execução do algoritmo, apenas a semântica de seus comandos.

3. Camada Avaliadora de Eficiência

Se for pedido que o aluno resolva um problema de ordenação usando um algoritmo específico como o *Couting Sort* que possui uma eficiência linear ($O(n)$) e o aluno implementar o algoritmo *Bubble Sort* com eficiência quadrática ($O(n^2)$), a ferramenta avaliadora poderá indicar uma falsa corretude na resolução do problema, já que os vetores resultantes serão idênticos ao final do processamento, pois os algoritmos tem a mesma eficácia, ou seja, termina com o vetor ordenado da mesma forma. Para evitar este resultado indesejado, propomos a Camada Avaliadora de Eficiência, responsável por analisar os algoritmos, classificá-los quanto a sua eficiência assintótica e compará-los, permitindo assim avaliar a corretude da solução do aluno de forma efetiva.

Esta camada terá acesso ao contador de passos básicos da ferramenta de programação, podendo contabilizar os comandos executados e a memória alocada para a execução de um algoritmo. A próxima seção detalha as técnicas propostas para a classificação automática de um algoritmo pelo Avaliador de Eficiência.

3.1. Arquitetura do Avaliador de Eficiência

O Avaliador de Eficiência será responsável por analisar os comandos executados pelo Interpretador de Comandos e gerar relatórios sobre a eficiência dos algoritmos implementados pelo usuário. A interface de comunicação entre estes componentes será intermediada por uma base de dados representada pela Tabela de Eficiência.

A Tabela de Eficiência contém as seguintes informações a serem registradas a cada execução: *id*: identificador único do algoritmo; *n*: dimensão da entrada de dados; $T(n)$: passos básicos acumulados na execução do Interpretador de Comandos; $M(n)$:

pico máximo de alocação de memória usado pelo algoritmo. O Avaliador de Eficiência deve gerar várias instâncias aleatórias de tamanhos *n* como entrada para o algoritmo e armazenar a quantidade de comandos básicos contabilizados pelo contador de comandos da ferramenta, alimentando a Tabela de Eficiência.

3.2. Interpolação Polinomial de Lagrange

A proposta de classificação assintótica por Interpolação Polinomial de Lagrange define um valor numérico que indicará o *coeficiente de classificação* (CC) da complexidade algoritmica. Dada uma função custo $T(n)$, o CC é um objeto matemático obtido através do cálculo da razão do valor desta função sobre sua derivada primeira. Propomos associar o CC a um esquema de classificação, gerado a partir de dois valores distintos α e β.

Estes valores não estão definidos explicitamente no intervalo de *n*, que representa as dimensões das instâncias tratadas pelos algoritmos, e devem ser escolhidos dentro deste intervalo para evitar a extrapolação polinomial no cálculo de $T(\alpha)$ e $T(\beta)$. Os valores numericos adotados neste trabalho foram empiricamente escolhidos para obter classificações mais estáveis. Os *fatores de classificação*(FC) representam as possíveis classificações assintóticas que a função $T(n)$ pode assumir. O esquema de classificação é realizada a partir de uma aproximação do *coeficiente de classificação* da função $T(n)$ comparadas com todos os *fatores de classificação* conhecidos..

Coeficiente de Classificação

Para identificarmos o *coeficiente de classificação* partindo dos princípios da Interpolação Polinomial de Lagrange é necessário uma equação com a função $f(n)$ e sua derivada primeira $f'(n)$ que posteriormente será comparado aos *fatores de classificação*, de modo a conseguirmos classificar a função custo que caracteriza o crescimento da função trabalhada. Para o cálculo do CC, utilizamos a equação definida em (1), para a qual atribuimos dois valores α e β como entrada, Desta forma conseguimos identificar o *coeficiente de classificação* a partir da função custo dada como entrada.

$$CC = \frac{f(\alpha)}{f'(\alpha)} - \frac{f(\beta)}{f'(\beta)}, \tag{1}$$

Fatores de Classificação

Para identificarmos os fatores de classificação, é necessário aplicar as expressões definidas na Tabela 1, estas expressões possuem demonstrações matemáticas que não são apresentadas por limitações no espaço deste trabalho. A coluna *Função* define nominalmente o tipo de função atribuída à expressão, a coluna *f(x)* define a função cujo FC será definido e a coluna *FC* define as expressões que usamos para identificar o *fator de classificação* para cada função elencada e que os algoritmos classifacados podem assumir tendo os pontos α e β fornecidos como entrada.

Definindo todos esses valores, podemos classificar a função custo $T(n)$ partindo da função interpoladora de Lagrange. Após a identificação dos *fatores de classificação*, o que nos resta é comparar o *coeficiente de classificação* de $T(n)$ entre todos os *fatores de classificação* encontrados.

Função	f(x)	FC
exponencial	ab^x	0
linear	ax	$\|\alpha - \beta\|$
quadrática	ax^2	$\|\frac{(\alpha-\beta)}{2}\|$
cúbica	ax^3	$\|\frac{(\alpha-\beta)}{3}\|$
logarítmica	$a \log_b x$	$\|\alpha ln\alpha - \beta ln\beta\|$
nlogn	$ax \log_b x$	$\|\frac{\alpha ln\alpha}{1+ln\alpha} - \frac{\beta ln\beta}{1+ln\beta}\|$
fatorial	$n!$	$\|ln\alpha - ln\beta\|$

Tabela 1. Funções $T(n)$ dos algoritmos clássicos calculados pelo componente medidor de eficiência do IGED.

3.3. Arquitetura Atual do IGED

Neste trabalho é proposta uma modificação na arquitetura do IGED ([Sousa Filho et al. 2012] et al.), com a finalidade de adicionar a funcionalidade de avaliação automática de eficiência de um algoritmo ao IGED.

Esta nova arquitetura é acrescida do componente Avaliador de Eficiência que alimenta sua Tabela de Eficiência através da integração com o componente Interpretador de Comandos, responsável por executar as instruções do algoritmo do usuário e agora servindo de contador dos passos básicos gerados pela tradução do algoritmo do aluno, realizada pelo compoenente Tradutor do IGED.

4. Resultados Computacionais

Para validarmos as metodologias de classificação de eficiência assintótica dos algoritmos, foram realizados experimentos computacionais sobre sete algoritmos clássicos.

Os sete algoritmos foram implementados na ferramenta e foram gerados 10 vetores aleatórios com dimensões crescentes, onde seus custos computacionais foram colhidos e armazenados na Tabela de Eficiência do Avaliador de Eficiência. Estes dados estão listados na Tabela 1e representam a função custo de processamento destes algoritmos.

4.1. Resultados Computacionais por Interpolação Polinomial de Lagrange

A Tabela 2 apresenta os *fatores de classificação* (FC) para α e β definidos por: $\alpha = 355$ e $\beta = 680$, sendo $\alpha = 2,55$ e $\beta = 7,85$ para os algoritmos *Fibonacci* e *Salesman*. A coluna *CC* indica os *coeficientes de classificação* e a coluna θ representa o limite assintótico firme de cada algoritmo cuja prova pode ser encontrada em Cormen [Cormen 2002]. Devemos observar que os *fatores de classificação* para os algoritmos *Fibonacci* e *Salesman* são os definidos a partir dos pontos $\alpha = 2,55$ e $\beta = 7,85$, logo, os *coeficiêntes de classificação de classificação* destes algoritmos deveram ser comparados a essa coluna especificamente, diferente dos demais que serão comparados aos *fatores de classificação* que foram definidos a partir dos pontos $\alpha = 355$ e $\beta = 680$.

A Tabela 3 apresenta os resultados referentes aos *coeficientes de classificação* resultantes da execução dos experimentos. A coluna Algoritmo apresenta os algoritmos estudados, a coluna CC apresenta os *coeficientes de classificação* resultantes da aplicação do método de classificação para cada algoritmo e a coluna θ representa a ordem de classificação assintótica após a comparação de cada FC com os CC resultantes, definindo a ordem de classificação assintótica.

θ	***FC***	
	$\alpha = 355; \beta = 680$	$\alpha = 2,55; \beta = 7,85$
2^n	0	0
n	325,0	5,3
n^2	162,5	2,65
n^3	108,3	1,76
$nlogn$	286,2	13,78
$logn$	2350,4	4,05
$n!$	0,64	1,12

Tabela 2. Valores numéricos dos *fatores de classificação* para $\alpha = 355$; $\beta = 680$ e $\alpha = 2,55$; $\beta = 7,55$.

Algoritmo	**Lagrange**	
	CC	θ
bubble	160,0	n^2
counting	327,4	n
merge	290,3	$nlogn$
quick	267,3	$nlogn$
binary	2050,1	$logn$
fibonacci	0,53	2^n
salesman	1,16	$n!$

Tabela 3. Valores numéricos dos *coeficientes de classficação* para o $T(n)$ dos algoritmos analisados.

Observa-se que os valores resultantes do CC de cada algoritmo são exatamente, quando comparados aos FCs, os valores definidos por cada classificaçãop assintótica θ.

5. Considerações Finais

Para validar a metodologia de classificação assintótica de eficiência dos algoritmos pelo Avaliador de Eficiência, que utiliza o modelo de classificação por Interpolação de Lagrange sobre as funções custo de processamento dos algoritmos, foram realizados experimentos sobre sete algoritmos clássicos de ordenação que possuem avaliações teóricas distintas. Os sete algoritmos foram classificados pelo Avaliador de Eficiência com a mesma avaliação assintótica firme (θ) encontrada na literatura, sendo esta uma avaliação teórica e a nossa experimental, demonstrando assim, a eficácia da proposta na classificação dos algoritmos. Diferente de outros trabalhos da literatura que avaliam apenas uma das características de um algoritmo, eficiência ou eficácia, o IGED permite avaliar as duas dimensões em consequência de seu controle sobre o Interpretador de Comandos.

Referências

Barbosa, M. A. C., Toscani, L. V., and Ribeiro, L. (2001). Anac - uma ferramenta para análise automática da complexidade de algoritmos. *Revista do CCEI*, 5(8):57–65.

Cormen, T. H. (2002). *Algoritmos: teoria e prática.* Campus, Rio de Janeiro, RJ.

Le Métayer, D. (1988). Ace: an automatic complexity evaluator. *ACM Trans. Program. Lang. Syst.*, 10(2):248–266.

Sousa Filho, G. F., Netto, D. P., Procópio, L. D. P., Formiga, A., and Brito, A. V. (2012). Tutor hipermídia baseado no modelo de autoria ncm para o interpretador gráfico de estrutura de dados. *XX Workshop sobre Educação em Computação no XXII Congresso da Sociedade Brasileira de Computação*.