Um Ambiente para Ensino de Programação com Avaliação Automática de Corretude e Eficiência

Gilberto F. Sousa Filho[1], Erick John F. Costa[1], Andrei A. Formiga[2], Alisson V. Brito[2]

[1]Departamento de Ciências Exatas – Universidade Federal da Paraíba (UFPB)
[2]Centro de Informática – Universidade Federal da Paraíba (UFPB).

{gilberto, erick.costa}@dce.ufpb.br, {andrei, 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.*

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. As duas formas de análise são importantes no trabalho com algoritmos, e por isso são enfatizadas em disciplinas de ensino de Algoritmos e Estruturas de Dados em cursos de Computação, Engenharias e outros. 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. Revisão Bibliográfica

Segundo Cormen (2002), analisar um algoritmo significa prever os recursos de que ele necessitará. Em geral, memória, largura de banda de comunicação ou hardware de computação são a preocupação primordial, 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.

Trabalhos Relacionados

O sistema ACE, *Automatic Complexity Evaluator* (Metayer 1988), é capaz de analisar algoritmos de propósito geral, entretanto implementados em linguagem

funcional. Partindo-se do programa inicial, uma função de complexidade de tempo é derivada. Quando o sistema realiza a conversão das equações a expressão final de complexidade será uma composição de funções bem conhecidas.

O ACME, Analisador de Complexidade Média (Silveira 1998), e o ANAC (Barbora 2001) são ferramentas para o cálculo da complexidade de algoritmos. Basicamente são programas de análise-síntese de códigos fontes escritos em uma linguagem procedural pré-estabelecida. Estes sistemas não avaliam a complexidade de algoritmos recursivos, já que os mesmos avaliam apenas a semântica de seus comandos.

Souza (2009) apresenta a ferramenta VisuAlg, um ambiente de programação que possui um compilador e interpretador para um pseudocódigo procedural e a funcionalidade de examinar a eficiência de um algoritmo, verificando quantas vezes uma determinada linha de código foi executada.

3. Camada Avaliadora

Sousa et al. (2012) apresentaram uma arquitetura para o Interpretador Gráfico de Estruturas de Dados (IGED), uma ferramenta que permite que alunos desenvolvam, numa linguagem própria, tarefas de programação passadas pelo professor, com a funcionalidade de animação das Estruturas de Dados utilizadas. Neste trabalho é proposta uma nova camada com a finalidade de realizar avaliação automática de corretude e eficiência de um algoritmo no IGED.

3.1 Componente Avaliador de Corretude

A Camada Avaliadora passa a ser dividida em três componentes: o primeiro componente é a Abstração de Estrutura de Dados, que contém a implementação de cada estrutura abordada e executará nestas estruturas os comandos recebidos e traduzidos pelo Interpretador de Comandos. Este componente trabalhará com duas instâncias idênticas da estrutura: a original e uma réplica. O objetivo da réplica é servir de controle da tarefa: o instrutor cadastrará um programa que serve de solução para a tarefa em questão; esse programa será executado sobre a réplica. Já a instância original sofrerá alterações na execução do programa inserido pelo aluno para resolver a tarefa.

O componente Avaliador de Eficácia terá acesso às duas instâncias (original e réplica) e comparará seus estados finais. Se os estados das estruturas estiverem iguais, a resolução da tarefa será considerada eficaz, caso contrário, será considerada falha.

3.2 Componente 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 criados pelo usuário. A interface de comunicação entre estes componentes será intermediada por uma base de dados representada na Figura 2 pela Tabela de Eficiência.

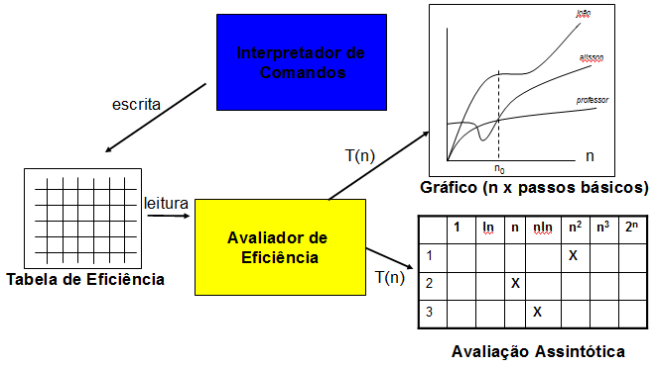

Figura 2. Arquitetura do Componente Avaliador de Eficiência

A Tabela de Eficiência contem as seguintes informações a serem registradas a cada execução: *id*, identificador único do algoritmo; *n*, dimensão da entrada de dados; *Proc(n)*, passos básicos acumulados na execução do Interpretador de Comandos; *M(n)*, pico máximo de alocação de memória usado pelo algoritmo.

A partir da Tabela de Eficiência são extraídos valores para a construção da função de custo do processamento *T(n)* de um algoritmo, com esta função o gráfico pode ser plotado e visualmente comparado com outros algoritmos pré-cadastrados, além disso sua classificação assintótica de eficiência pode ser gerada a partir de um procedimento matemático proposto a seguir.

3.3 Avaliação Assintótica

Propomos uma classificação assintótica, ou seja, por semelhança da função custo *T(n)* do algoritmo do usuário por meio do cálculo da Correlação de Pearson desta com as funções clássicas de avaliação teórica $F(n) = \{c, c \cdot log\ n, c \cdot n, c \cdot n log\ n, c \cdot n^2, c \cdot n^3, c \cdot 2^n, c \cdot n!\}$, sendo *c* o coeficiente constante de cada função. Logo, a função *F(n)* que tiver o melhor coeficiente de Correlação de Pearson passa a ser considerada a função que classifica a eficiência deste algoritmo.

Correlação de Pearson

O coeficiente de correlação de Pearson é uma medida estatística que representa o grau de dependência entre duas variáveis aleatórias. Normalmente representado pela letra grega ρ, a correlação entre variáveis X e Y é dada pela fórmula $\rho = Cov(X, Y) / (Var(X)\ Var(Y))^{1/2}$, sendo Cov(X, Y) a covariância entre X e Y, e Var é a variância (Shao, 2003). Dados dois conjuntos de amostras de X e Y, a correlação pode ser entendida como o ângulo entre os vetores formados pelas amostras de X e Y:

- Se ρ = 1, o ângulo α = 0°, *X* e *Y* são colineares (paralelos). Significa uma correlação perfeita positiva entre as duas variáveis;
- Se ρ = 0, o ângulo α = 90°, *X* e *Y* são ortogonais. Significa que as duas variáveis não dependem linearmente uma da outra;
- Se ρ = -1, o ângulo α = 180°, *X* e *Y* são colineares com sentidos opostos. Significa uma correlação negativa entre as variáveis, ou seja, se uma aumenta a outra diminui.

A partir desta técnica podemos comparar a eficiência de duas soluções distintas, permitindo assim descobrir se a solução fornecida pelo usuário está classificada na mesma eficiência da solução do tutor/autor da atividade.

4. Resultados Experimentais

Para avaliarmos a metodologia de classificação de eficiência assintótica dos algoritmos do usuário proposta na Seção 3.3, foram realizados alguns experimentos computacionais sobre quatro algoritmos clássicos de ordenação: *Bubble Sort*, *Counting Sort*, *Merge Sort* e *Quick Sort*. Estes algoritmos foram implementados na ferramenta IGED e 10 vetores aleatórios com diferentes dimensões foram aplicados aos algoritmos, sendo os dados armazenados pelo Avaliador de Eficiência.

A Tabela 1 apresenta os valores numéricos, gerados pelo Avaliador de Eficiência, na aplicação da Correlação de Pearson ao vetor *T(n)* de cada algoritmo e os vetores das funções assintóticas clássicas. A coluna *T(n)* representa a instância do algoritmo cujo vetor será comparado com as funções assintóticas clássicas, a coluna **Θ** representa o limite assintótico firme de cada algoritmo cuja prova pode ser encontrada em Cormen (2002). A coluna *F(n)* representa as funções clássicas abordadas na avaliação assintótica, sendo esta coluna subdividida em colunas para cada função,

representando o valor da Correlação de Pearson destas funções com a função *T(n)* de cada algoritmo analisado.

Tabela 1. Valores de Correlação entre as funções T(n) dos algoritmos clássicos de ordenação com funções assintóticas clássicas (*c=1*).

T(n)	Θ	F(n)							
		c	c·log n	c·n	c·n·log n	$c \cdot n^2$	$c \cdot n^3$	$c \cdot 2^n$	c·n!
Bubble	n^2	0	0,875583	0,97989	0,98659	**0,99967**	0,982987	0	0
Counting	n	0	0,951869	**1**	0,999269	0,974446	0,92823	0	0
Merge	n·log n	0	0,941888	0,999521	**0,999963**	0,980708	0,938316	0	0
Quick	n·log n	0	0.932647	0.997502	**0.998760**	0.982690	0.941243	0	0

Como apresentado na Seção 3, o melhor valor que a Correlação de Pearson deve assumir é 1; logo, podemos observar que os melhores valores atingidos por cada algoritmo em relação às funções *F(n)* candidatas são exatamente as funções definidas na coluna Θ, limite teórico firme. Como exemplo podemos observar o algoritmo Merge Sort, que teve o maior valor de Correlação de Pearson igual a 0,999963 para a função $F(n) = c \cdot n \cdot \log n$, sendo sua avaliação assintótica firme $\Theta(n \cdot \log n)$.

5. Considerações Finais

Para validar a metodologia de classificação assintótica de eficiência dos algoritmos pelo Avaliador de Eficiência, que utiliza uma análise da Correlação de Pearson sobre as funções custo de processamento dos algoritmos, foram realizados experimentos sobre quatro algoritmos clássicos de ordenação que possuem avaliações teóricas distintas. Os quatro 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, sendo assim, uma ferramenta mais eficaz na avaliação automática das tarefas propostas.

Referências Bibliográficas

Barbosa, Marco A. C.; Toscani, L. V.; Ribeiro, L. (2001) ANAC - uma ferramenta para análise automática da complexidade de algoritmos. Revista do CCEI, Bagé: Universidade da Região da Campanha - URCAMP, v. 5, n. 8, p. 57-65.

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

Metayer, D. L. (1988) Ace: An automatic complexity evaluator. ACM Transactions on Programming Languages and Systems, New York, v.10, n.2, p.248-266.

Shao; Jun (2003) Mathematical Statistics. Springer Science, New York, NY.

Silveira; Carlos M. D. (1998) Analisador de Complexidade Média Baseado nas Estruturas Algorítmicas. Pelotas: UFPEL.

Sousa Filho, G. F.; Netto, D. P.; Procópio L. D. P.; Formiga, A.; Brito, A. V. (2012) "Tutor hipermídia baseado no modelo de autoria NCM para o Interpretador Gráfico de Estrutura de Dados". In: XX WEI no XXII CSBC. Curitiba.

Souza, C. M. (2009) VisuAlg - Ferramenta de Apoio ao Ensino de Programação. Revista TECCEN, v. 2, n. 2, ISSN 1984-0993.