UFMG/ICEx/DCC — Algoritmos e Estrutura de Dados 3

Trabalho Prático 3

Bacharelado — 1º Semestre de 2012

Este trabalho prático tem por objetivo exercitar a implementação de paradigmas computacionais. Focaremos no paradigma de *Programação Dinâmica* com o intuito de analisar o seu desempenho em termos de tempo de execução e a quantidade de memória alocada.

1 Definição do Problema

Este trabalho aborda o problema da contagem das possíveis formas de colocar parênteses em uma expressão booleana. Os parênteses possuem papel crucial no resultado de uma expressão sujeita à operadores que atuam sobre suas variáveis. A posição dos parênteses determina sobre quais sub-expressões uma operação será aplicada, de modo que define as precedências a serem respeitadas durante a computação do resultado final da expressão.

Uma expressão booleana é definida por variáveis booleanas (**Var**), operadores (**Op**) e sub-expressões (**Exp**). A gramática abaixo descreve cada um desses componentes:

$$\textbf{Exp} \rightarrow (\textbf{Exp Op Exp}) \mid (\text{not } \textbf{Exp}) \mid \textbf{Var}$$
$$\textbf{Op} \rightarrow \text{and} \mid \text{or} \mid \text{xor} \mid \text{nand} \mid \text{imp} \mid \text{eqv}$$
$$\textbf{Var} \rightarrow \text{True} \mid \text{False}$$

Observe que a presença dos parênteses é de suma importância para definir sub-expressões. Dessa maneira, exigiremos que eles estejam presentes sempre que houver qualquer formulação semelhante as apresentadas na gramática acima. Mesmo casos triviais, em que a atuação dos parênteses é implícita, como em $Exp_1 = (True\ and\ False)$, a presença deles será obrigatória.

Neste trabalho, o aluno irá contabilizar as formas de colocar parênteses em uma dada expressão booleana Exp_i de modo que resulte em *True*. Inicialmente, Exp_i possui t termos e não possui parênteses algum. A seguir, um exemplo simples que ilustra o objetivo do trabalho, através da expressão dada Exp_1 (com 6 termos):

$$Exp_1 = \text{not False or False and True}$$

Formulação possível de Exp_1 = True:

$$Exp_1 = ((\text{not False}) \text{ or } (\text{False and True}))$$
$$Exp_1 = (\text{True or False})$$
$$Exp_1 = \text{True}$$

No exemplo acima, é apresentada uma formulação possível de atribuir parênteses à Exp_1 de modo que o resultado seja verdadeiro. Entretanto, há outras maneiras de se alcançar esse objetivo. O foco do trabalho é justamente contabilizar quantas formulações distintas de uma dada expressão resultam em *True*.

Portanto, o trabalho prático 3 consiste em solucionar o problema descrito através do paradigma computacional de *Programação Dinâmica*. O aluno **deve** apresentar a fórmula recursiva do seu algoritmo, bem como uma análise experimental com o intuito de avaliar:

1. Desempenho do algoritmo (tempo de execução)* em função do número t de termos da expressão booleana de entrada;
2. Quantidade de memória alocada (número de bytes) durante a execução em função do número t de termos da expressão booleana de entrada.

*O tempo de execução do algoritmo implementado **deve** ser mensurado pela função ***gettimeofday()***.
A implementação dos algoritmos **DEVE** utilizar alocação dinâmica de memória.

1.1 Entrada e Saída

O arquivo executável deve ser chamado de *tp3* e deve receber como parâmetro o arquivo de entrada *input.txt* e o arquivo de saída *output.txt*, conforme demonstrado a seguir:

```
./tp3 -i input.txt -o output.txt
```

O arquivo *input.txt* deve conter as seguintes informações, na ordem indicada abaixo:

1. número i de instâncias do problema;
2. i expressões booleanas sem parênteses, uma por linha.

Exemplo de entrada genérico (arquivo *input.txt*):

2 // Número i de instâncias not False or False and True // Expressão booleana Exp_1 da instância 1 True and not True and True // Expressão booleana Exp_2 da instância 2

Cada expressão booleana é definida por um conjunto de t termos. Os testes de correção irão considerar $0 < t \leq 50$. Garanta que seu algoritmo funcione para esses valores de t.

A saída do programa deve ser impressa no arquivo *output.txt* e deve conter o número de formas de atribuir parênteses à cada expressão booleana de entrada, de modo que o resultado obtido seja *True*. Os valores impressos no arquivo de saída devem seguir a mesma ordem apresentada no arquivo de entrada. Dessa maneira, cada linha l do arquivo de saída, refere-se à expressão booleana da instância l apresentada no arquivo de entrada.

Exemplo de saída genérico (arquivo *output.txt*):

5 // 5 formas de atribuir parênteses à Exp_1 da instância 1 para obter resultado *True* 0 // Zero formas de atribuir parênteses à Exp_2 da isntância 2 para obter resultado *True*

Os comentários nos exemplos de entrada e saída foram apresentados apenas por propósitos didáticos, logo eles **NÃO DEVEM** ser incluídos nos arquivos para submissão. Além disso, entrada e saída padrão devem seguir rigorosamente o formato descrito, inclusive no caso de múltiplas instâncias. Sendo assim, em ambos os arquivos, dados de diferentes instâncias devem ser apresentados na mesma ordem e em sequência (na linha seguinte do último dado da instância anterior). Instâncias distintas para o problema devem ser geradas pelo próprio aluno para testar e avaliar seu algoritmo.

2 O que deve ser entregue:

2.1 Documentação: deve abranger pelo menos os seguintes pontos

- Introdução do problema apresentado.
- Modelagem e solução do problema: explique o paradigma utilizado para resolver o problema, tal como sua respectiva complexidade de tempo e espaço.
- Principais decisões de implementação.
- Análise experimental quanto aos critérios especificados no trabalho.
- A documentação **não** pode exceder 10 páginas.

2.2 Código:

- O código fonte do trabalho deve ser submetido para compilação e execução em ambiente Linux, tendo como padrão os computadores dos laboratórios de graduação do DCC;
- Deve ser **obrigatoriamente** escrito na linguagem C (trabalhos implementados em outras linguagens como C++/Java/Python e outras **não** serão aceitos);

- As estruturas de dados devem ser alocadas dinamicamente e o código deve ser modularizado (ou seja, dividido em múltiplos arquivos fonte e fazendo uso de arquivos cabeçalho - .h);
- O utilitário Make deve ser utilizado para compilar o programa **(o arquivo de makefile deve ser submetido juntamente com o código fonte)**;
- A saída deve ser impressa seguindo estritamente o formato da especificação, caso contrário o resultado será considerado errado;
- O arquivo executável deve ser chamado de **tp3**. Não serão aceitos outros nomes de executáveis além do mencionado;
- Faça seu código de forma legível.

2.3 Entrega:

- Data de entrega: 08/05/2012
- Submissão: a documentação e o código do trabalho devem ser submetidos ao *minha.ufmg*. Para isso, compacte os dois (formato *tp3_NomeSobrenome.zip*) e faça a submissão. Teste seu arquivo compactado antes de enviá-lo.
- Apenas a documentação deve ser entregue impressa na secretaria do DCC. Não coloque nos escaninhos dos professores, entregue para a secretária para que sua documentação seja colocada no envelope de AEDS3. A documentação impressa pode ser entregue no dia útil seguinte da submissão digital. **Trabalhos que não tiverem a documentação entregue na secretaria, dentro do prazo de entrega, receberao nota 0.**
- Será postada uma planilha no Moodle sobre a entrevista do trabalho, leia-a e siga as orientações para o agendamento da sua entrevista.
- Será adotado média harmônica entre a pontuação obtida na execução e na documentação do TP, o que implica em valor zero caso alguma das partes não seja apresentada.
- A política para desconto por atraso de entrega do trabalho prático considera a fórmula:

$$\frac{2^{d-1}}{0.32}\%$$

onde d é o atraso em dias úteis. Note que após 5 dias úteis, o trabalho não pode ser mais entregue.

Observações

- O formato de submissão **DEVE** ser respeitado: *tp3_NomeSobrenome.zip*. Por favor, **NÃO** submetam trabalhos em outros formatos (.rar, .tar, .tar.gz). Certifique-se que seu nome completo foi incluído no nome do arquivo.
- **EXCLUA** o arquivo executável do arquivo compactado a ser submetido no *minha.ufmg*.
- **INCLUA** seu email na capa da documentação.

Referências

Projeto de Algoritmos com implementação em Pascal e C. Nívio Ziviani - *http://www.dcc.ufmg.br/algoritmos/*.

Introduction to Algorithms. T. Cormen, C. Leiserson, R. Rivest, and C. Stein. MIT Press, 3rd edition, 2009.

Introdução aos Fundamentos da Computação. Newton Vieira. Pioneira Thomson Learning, 2006.