Trabalho Prático 02 – AEDS 1 – EM TRIOS

Professora: Thais R. M. Braga Silva

Valor: 10 pontos

Data de Entrega: 11/04/2021

Forma de Entrega: PVANet (formato .doc, .pdf, .zip ou .tar.gz)

O objetivo deste trabalho prático é permitir a avaliação do impacto causado pelo desempenho dos algoritmos em sua execução real. Vimos em sala de aula que existem problemas e algoritmos de complexidade exponencial, chamados de intratáveis. Nesses casos, os programas, ao serem executados podem demorar uma quantidade de tempo não razoável para encontrar uma solução, dependendo do tamanho da entrada. Vamos observar, portanto, como isso ocorre na prática. Para tanto, cada grupo fará uma implementação para o "Problema da Satisfabilidade (SAT)". Mais especificamente, trataremos do problema conhecido por 3-FNC-SAT, que é uma versão de SAT. Esse é um problema intratável, pois sua solução exata somente é possível através do cálculo e avaliação de todas as possíveis saídas, o que chamamos de força bruta. Em seguida, essa implementação deverá ser executada para diferentes valores de entrada N, e o tempo gasto para que o programa termine, em cada caso, deverá ser medido por meio de comandos do sistema operacional.

Em linhas gerais, 3-FNC-SAT é um problema que recebe como entrada uma equação booleana contendo X cláusulas, cada uma delas composta por exatamente 3 literais. Por estar na Forma Normal Conjuntiva (FNC), os literais de cada cláusula estão ligados pelo operador de disjunção (|) e as cláusulas em si estão ligadas umas às outras pelo operador de conjunção (&). Cada literal é uma variável booleana ou uma negação (!) de uma variável booleana. Exemplo: E = (x1 | !x3 | x4) & (!x1 | x2 | x5). A equação E é uma possível entrada para o problema 3-FNC-SAT. Cada parênteses representa uma cláusula. Cada x_i ou $!x_i$ representa um literal. O objetivo deste problema é encontrar uma configuração de valores lógicos para as variáveis envolvidas na equação de entrada, tal que a mesma seja avaliada como verdadeira (isto é, tal que ela seja satisfeita). No exemplo acima, a solução x1 = V, x2 = V, x3 = F, x4 = F e x5 = F, isto é, o conjunto R = {V,V,F,F,F}, satisfaz a equação. Note que ela não é a única solução possível.

Existem diversas possíveis implementações para o Problema 3-FNC-SAT. Entretanto, as mais utilizadas requerem recursos de programação que ainda não foram estudados por vocês. Dessa forma, adotaremos uma estratégia mais direta, visto que o objetivo principal do trabalho é a avaliação de desempenho, e não o desenvolvimento do algoritmo. Cada grupo deverá implementar um programa, em linguagem C, para o problema 3-FNC-SAT da seguinte forma:

- Procurar na Web um algoritmo que gere todas as combinações possíveis para N valores lógicos. Exemplo: se N = 2, o algoritmo deve gerar {V,V}, {V,F}, {F,V}, {F,F}. Você deverá utilizar a linguagem C. Portanto, se o algoritmo encontrado estiver em outra linguagem de programação, vocês deverão convertê-lo para linguagem C. Veja que o algoritmo deve ser parametrizado, isto é, capaz de gerar todas as combinações para um valor N qualquer.

- Seu programa deverá funcionar em dois modos: interativo e automático. No modo interativo, o programa deverá receber do usuário a equação booleana E e então **procurar por todos os conjuntos de valores booleanos R que a satisfaça, se houver**. Para tanto, o usuário deverá passar como entrada o número C de cláusulas, o número N de variáveis lógicas e, para cada cláusula, indicar o trio de variáveis lógicas que a compõem, bem como o estado (negado ou não) de cada uma delas. O formato padrão para essa última entrada, para cada cláusula i, é: c_i = {(1<=v<=N,[1|2]),((1<=v<=N,[1|2]),((1<=v<=N,[1|2])}. Cada parêntesis representa uma tupla que contém o número de uma variável e o estado dela, sendo que o valor 1 indica que a variável está negada e 2 para não negada. Para a equação E apresentada como exemplo acima, a entrada seria C = 2, N = 5 e, para a cláusula 1, os valores c1 = {(1,2),(3,1),(4,2)} indicando que nela são utilizados x1 não negado, x3 negado e x4 não negado. Para a cláusula 2 seriam os valores c2 = {(1,1),(2,2),(5,2)}. Em seguida, o programa começa a gerar todas as possíveis combinações de valores booleanos de tamanho N. Para cada combinação gerada, testa se a mesma satisfaz a equação da entrada. Se satisfizer, apresenta-a na saída. Caso contrário, se ainda houver, gera próxima combinação possível, e procede com o mesmo comportamento.

- No modo automático, o valor de N deverá ser atribuído conforme solicitado para os testes abaixo. O número de cláusulas C deverá ser (N/3)*2. Em seguida, seu programa deverá gerar, aleatoriamente, a expressão booleana. Para essa expressão, procurar todos os conjuntos de valores booleanos que a satisfaça e apresentá-los, se houver. Para gerar aleatoriamente a expressão, seu programa deverá declarar uma matriz CxN, onde cada linha representa uma cláusula. Inicialmente, a matriz deve ser preenchida com zeros. Em seguida, para cada linha, devem ser escolhidas 3 colunas, gerando 3 número aleatórios entre 0 e N-1 (esses números indicam as variáveis escolhidas para a cláusula). Para cada um desses números, escolher aleatoriamente entre os valores 1 (negado) e 2 (não negado) e preenchê-lo na posição correspondente na matriz. Em seguida, proceder para a geração de todas as combinações de valores booleanos de tamanho N, procedendo da mesma forma como descrito para o modo interativo (OBSERVAÇÃO: o número de variáveis usado na equação pode, na verdade, ser menor do que N. Nesse

caso, verificar quantas e quais variáveis foram efetivamente utilizadas e gerar todas as combinações possíveis para as mesmas). Abaixo um exemplo de uma matriz preenchida para representar uma equação E com C=4 e N=6. Essa matriz representa E = (x2 | !x3 | ! x4) & (!x1 | !x2 | x6) & (x3 | x4 | x5) & (x1 | x3 | x6) .

0	2	1	1	0	0
1	1	0	0	0	2
0	0	2	2	2	0
2	0	2	0	0	2

- Execute o programa no modo automático para os N= 15, 20, 30 e 40. Ao executar o programa, utilize uma ferramenta para medição do tempo de execução, como o comando *time* do Unix. Você deve medir o tempo que foi gasto para encontrar a resposta (todos os conjuntos booleanos de tamanho N que satisfazem a equação E), se houver.

- Faça um relatório final contendo uma breve explicação do código implementado, indicando como funciona o algoritmo de geração de combinações utilizado, de onde foi obtido, como o seu programa recebe as entradas, avalia a expressão booleana para cada combinação e mostra a resposta. O relatório também deverá conter as configurações de hardware e software da máquina utilizada para realização dos testes e os resultados dos tempos de execução para os valores de N indicados acima. Você poderá utilizar gráficos e/ou tabelas para apresentá-los.

- Responda também no relatório à seguinte pergunta: seria razoável executar o seu algoritmo para valores de N maiores do que 45? Justifique a resposta.

- Um arquivo contendo o código fonte produzido e o relatório final (pode ser um .doc, .pdf, .zip, .rar ou .tar.gz) deverá ser entregue até a data limite através do PVANet. Lembrem-se de colocar os nomes e matrículas dos integrantes do grupo.

Em particular, atente para:

- O programa deve estar bem organizado, de acordo com as práticas de projeto e implementação vistas até aqui na disciplina

- O programa deve estar bem indentado e comentado

- Caso apareçam números fixos no código, estes devem ser definidos como constantes

- **Trabalhos copiados serão penalizados.**

ATENÇÃO: Soluções que não correspondam à implementação descrita acima serão duramente penalizadas por não atenderem a especificação.

Como será a avaliação:

- Entrevista com o monitor: o monitor avaliará a compilação e execução do trabalho, bem como conduzirá alguns testes sobre o código entregue. Perguntas serão feitas também sobre as decisões de projeto e detalhes de implementação;
- Avaliação da professora: a professora avaliará o código e a documentação entregue e, juntamente com a avaliação fornecida pelo monitor, deliberará sobre a nota final de cada grupo.