MAC0239: Exercício-Programa 1

Versão 1

26 de setembro de 2022

1 Introdução

Mastermind é um jogo para dois jogadores onde um deles assume o papel de criador do código (*codemaker*) e o outro tentará decifrá-lo (*codebreaker*). O código é formador colocando pinos de uma das 6 em cores em cada um dos 4 espaços apropriados. Depois de criado o código, o *codebreaker* tem um número pré-determinado de tentativas para adivinhar o código gerado. A cada tentativa, o *codemaker* indicará por meio de pinos o quão próximo da resposta o chute do *codebreaker* está: para cada pino da cor correta na posição certa, um pino de cor preta de feedback é dado, e para cada pino de uma cor certa, mas em um posição incorreta é fornecido um pino de cor branca.

Mais informações sobre o jogo podem ser vistas na página da Wikipedia[1]. Também há uma versão do jogo para navegador[2].

[1] `https://en.wikipedia.org/wiki/Mastermind_(board_game)`
[2] `http://www.archimedes-lab.org/mastermind.html`.

2 Modificações ao Mastermind Original

Neste exercício-programa propomos uma versão simplificada do Mastermind, no que diz respeito à natureza do feedback: ao contrário do jogo original, cada posição do feedback indica se a cor naquela posição está correta ou não. Além disso, em vez trabalharmos com somente 6 cores e 4 espaços, vamos usar um número c de cores e um número n de espaços que serão fornecidos como argumento para o programa.

3 Modelagem

Para facilitar a modelagem (e simplificar o código), vamos indicar os espaços por naturais de 0 até $n-1$ e as cores por naturais de 0 até $c-1$. Assim, podemos usar variáveis proposicionais da forma x_{ij}, onde $v(x_{ij}) = 1$ se o pino no espaço i é da cor j.

4 Formato de Entrada

A maioria dos resolvedores SAT aceita entrada no formato "DIMACS CNF", que é um padrão de texto simples na forma normal clausal (CNF). Toda linha iniciada por "c"é um comentário. A primeira linha não-comentário deve ser da forma:

p cnf NÚMERO_DE_VARIÀVEIS NÚMERO_DE_CLÁUSULAS

Cada linha não comentada após esta define uma cláusula. Cada uma dessas linhas é uma lista de inteiros separados por espaço em branco. Um valor positivo indica que o literal correspondente à variável (assim, 4 indica o literal p_4), e um valor negativo indica a negação desta variável (assim, -5 indica o literal $\neg p_5$). Cada linha deve terminar com o número 0 seguido ou não de espaço.

Assim, a fórmula

$$(p_1 \vee \neg p_5 \vee p_4) \wedge (\neg p_1 \vee p_5 \vee p_3 \vee p_4) \wedge (\neg p_3 \vee \neg p_4)$$

pode ser expressa com o seguinte arquivo `exemplo.cnf`

```
c Esse é um comentário
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

A linha iniciada por "p cnf" indica que esse é um problema SAT na forma normal clausal/conjuntiva, com 5 variáveis e 3 cláusulas.

Mais informações sobre isso no endereço `http://www.satcompetition.org/2004/format-solvers2004.html` .

5 Código

5.1 Mastermind em C

Antes de iniciar esse trabalho, você deverá baixar os fontes do programa `minisat`, compilar e instalar este programa em algum um lugar conveniente da sua máquina, que deverá ser informado para o programa em C modificar. Mais informações sobre isso adiante.

Junto com este enunciado você vai encontrar o arquivo `MastermindEmC.zip`, o qual deverá ser descompactado e modificado por você neste exercício-programa conforme as instruções a seguir. Para compilar esse programa com diversos arquivos estamos fornecendo um arquivo Makefile, o qual você deverá invocaram usando o comando `make`. Esta configuração tem o intuito de fazer você e se acostumando a modificar programas desenvolvidos por outras pessoas e a utilizar ferramentas de produtividade em programação como `make`.

O programa executável gerado pelo `make` contém uma simulação do jogo Mastermind; neste código vocês devem alterar **somente** as funções `convert_feedback` (no arquivo `codebreaker.c`) e `codemaker_feedback` (no arquivo `codemaker.c`).

A função `convert_feedback` traduz o feedback em fórmulas para o SAT solver. Essa função representa o que o *codebreaker* "aprende"com o feedback dado pelo *codemaker*. Caso todas as atribuições estejam corretas, ela termina devolvendo `True`, senão executa a função de estratégia dada (observe os comentários no arquivo `codebreaker.c`).

A função `codemaker_feedback` codifica o resultado do último chute. Na versão apresentada, que deverá ser alterada, a função devolve um vetor com o mesmo número de posições que o código, de tal forma que contém 1 seu último chute acertou a cor naquele lugar; e devolve 0 caso contrário.

O programa executável `mastermind` recebe 3 parâmetros, nesta ordem: número de espaços, número de cores e número de tentativas. Assim, a seguinte chamada: `python3 mastermind.py 4 6 10`, indica uma rodada do Mastermind com 4 espaços, 6 cores e 10 tentativas.

O programa imprime o código gerador pelo *codemaker* e cada uma das tentativas feitas pelo *codebreaker*. No final, ele imprime quem foi o vencedor.

5.2 Modificações solicitadas

Você deverá realizar as seguintes modificações no seu programa.

1. No arquivo `codebreaker.h` você deverá alterar a constante (`#define`) `SOLVER_CALL` para que ela contenha o caminho até a versão executável do `minisat` conforme instalado no seu computador.

2. No arquivo `codemaker.c`, você deverá alterar a função `codemaker_feedback` para que ela indique também cores presentes em algum espaço, porém Inserida na posição errada. Por exemplo, se o código for [0 1 2 3] e a última tentativa foi [0 2 4 6], o feedback devolvido poderá ser [1 2 0 0], com a seguinte interpretação. O número 1 indica que a primeira posição está com a cor 0 correta; o número 2 na segunda posição indica que a cor presente nessa posição ocorre em outra posição do código; por fim, o número 0 nas duas últimas posições indica que aquelas cores não ocorrem em qualquer lugar do código.

3. No arquivo `codebreaker.c`, você deverá alterar a função `convert_feedback`, de modo a utilizar o feedback mais informativo gerado pela modificação anterior.

Você pode inserir quantas funções auxiliares achar conveniente, contanto que faça as modificações correspondentes nos arquivos de cabeçalho `.h`.

6 Experimentos

Além de implementar as modificações acima, vocês devem realizar o seguinte experimento: para cada $n \in \{4, 16, 64\}$ e cada $c \in \{6, 36, 108\}$, execute 10 vezes o Mastermind com $2c$ tentativas. Anote em uma tabela o número de vezes em que o CodeBreaker ganhou, em outra tabela indique o número médio de tentativas que o CodeBreaker precisou para ganhar.

Repita o mesmo experimento substituindo sua estratégia pela estratégia inicial oferecida. Você nota alguma diferença?

Escreva um breve relatório contendo: a descrição da nova estratégia implementada, diga quantas cláusulas sua estratégia produz e qual o tamanho delas em termos de n e c, as tabelas com os resultados dos experimentos para cada estratégia e uma conclusão comparando as estratégias.

7 Entrega

Entregue o código com sua versão modificada do exercício-programa e o relatório em PDF em um arquivo zip no campo adequado no e-disciplinas. O nome do arquivo zip deve seguir a seguinte regra: se seu nome for Fulano Ciclano Beltrano, o nome do arquivo deve ser: `ep1_fulano_ciclano_beltrano.zip`.