Trabalho 1 – Resolvedor de Sudoku$_N$ Multithread

INE5410 – Programação Concorrente – UFSC
Prof. Márcio Castro

1 Introdução

O Sudoku é um quebra-cabeça baseado na colocação lógica de números criado por *Howard Garns*. Ele consiste em uma grade de tamanho 9x9, constituída por 9 subgrades de tamanho 3x3 denominadas **regiões**. Inicialmente, as células da grade podem estar vazias ou numeradas de 1 à 9. Uma **unidade** representa um conjunto de células de uma *linha*, *coluna* ou *região* da grade. Portanto, cada célula pertence a exatamente à três unidades distintas.

O objetivo do quebra-cabeça é preencher cada célula vazia com um número de 1 à 9 de forma que cada *unidade* contenha todos os números de 1 à 9. Em outras palavras, uma solução correta contém cada célula vazia preenchida com números de forma que nenhuma *linha*, *coluna* ou *região* da grade contenha o mesmo número mais de uma vez. Normalmente, um quebra-cabeça Sudoku contém somente uma solução possível. Figura 1a apresenta um exemplo de um quebra-cabeça Sudoku com uma região marcada em azul e com as três unidades em amarelo da célula marcada com um círculo vermelho. Figura 1b mostra a solução para o quebra-cabeça da Figura 1a, com os números inseridos nas células vazias apresentados em verde.

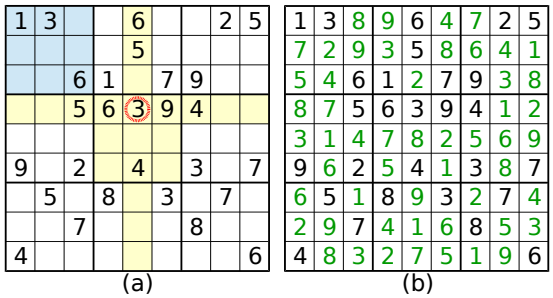

Figura 1: (a) Um quebra-cabeça Sudoku. (b) A solução para o quebra-cabeça.

Resolver um quebra-cabeça Sudoku clássico, como mostrado na Figura 1a, pode não ser uma tarefa muito simples para uma pessoa (ela pode levar vários minutos para encontrar a solução). Porém, um programa de computador pode encontrar a solução em apenas alguns milissegundos com uso de força bruta (i.e., um programa que teste todas as combinações possíveis até encontrar a solução correta).

Para deixar essa tarefa mais difícil, vamos considerar um modelo mais genérico para o quebra-cabeça Sudoku denominado Sudoku$_N$. Nesse modelo, vamos considerar que N corresponde ao número de células de uma linha/coluna em uma grade quadrada. Além disso, vamos considerar que cada grade possui N **regiões** e que cada região possui N células. A versão clássica do cabra-cabeças Sudoku possui essas propriedades, ou seja, ela representa uma instância desse modelo genérico com $N = 9$ (i.e., Sudoku$_9$ possui uma grade de 9x9, com 9 regiões e 9 células em cada linha/coluna/região). Com esse modelo, podemos considerar qualquer grade de tamanho NxN, onde $N = n^2$ para $n = 3, 4, 5, \ldots$, como por exemplo 9x9, 16x16, 25x25 . . . , com células podendo assumir valores entre 1 e N. Para ficar ainda mais divertido, vamos considerar que a grade de entrada possui um estado inicial que permite chegar a inúmeras diferentes soluções para o quebra-cabeça.

2 Definição do Trabalho

O trabalho consiste em desenvolver um programa utilizando POSIX threads (Pthreads) que encontra o número de soluções possíveis para um dado quebra-cabeça Sudoku$_N$ de entrada. O arquivo de entrada contendo o quebra-cabeça consiste em uma lista de números inteiros separados por espaço (`' '`) ou nova linha (`'\n'`). O primeiro inteiro representa o tamanho da região (n) seguido por n^4 inteiros contendo os valores da grade (vamos assumir $n \leq 8$). O valor de cada célula do quebra-cabeça de entrada poderá ser zero (representando uma célula vazia) ou qualquer valor entre 1 e N (representando uma célula preenchida). Os valores das células são dispostos da esquerda para a direita, de cima para baixo. Para facilitar a sua tarefa, está disponível no Moodle um programa em C que lê um quebra-cabeça Sudoku$_N$ de um arquivo e o armazena em um vetor de tamanho n^4.

O seu programa deverá receber um parâmetro pela linha de comando referente ao número de *threads* a serem utilizadas na computação. Após serem criadas, as *threads* deverão dividir o trabalho de encontrar as soluções do quebra-cabeça Sudoku$_N$ fornecido. Tenha em mente que é preciso evitar ao máximo que *threads* permaneçam ociosas sem realizar nenhum trabalho. Abaixo são mostrados dois arquivos de entrada de exemplo. A Figura 2 contém um quebra-cabeça Sudoku$_9$ ao passo que a Figura 3 contém um quebra-cabeça Sudoku$_{16}$. O número de soluções corretas em cada um dos exemplos é 1 e 300.064, respectivamente.

```
3
1 3 0 0 6 0 0 2 5
0 0 0 0 5 0 0 0 0
0 0 6 1 0 7 9 0 0
0 0 5 6 3 9 4 0 0
0 0 0 0 0 0 0 0 0
9 0 2 0 4 0 3 0 7
0 5 0 8 0 3 0 7 0
0 0 7 0 0 0 8 0 0
4 0 0 0 0 0 0 0 6
```

Figura 2: Exemplo de um Sudoku$_9$.

```
4
0 0 0 15 9 0 16 0 0 6 0 0 13 7 0 0
0 13 6 0 0 15 0 0 7 0 0 4 8 0 0 0
2 0 0 0 0 0 0 13 15 0 10 0 0 3 9 0
7 0 5 0 6 3 0 0 0 12 0 0 0 0 14 11
0 5 0 0 0 11 8 0 0 0 0 15 3 0 0 0
0 6 0 1 10 0 5 0 11 0 2 0 12 14 0 0
0 0 16 0 0 0 0 3 12 0 0 8 0 6 7 0
3 0 0 10 0 6 0 0 0 13 1 0 0 0 0 8
12 0 0 0 0 5 2 0 0 0 8 0 0 15 1 0
0 0 1 9 4 0 0 0 0 15 0 0 7 0 0 0
0 8 2 0 3 0 0 16 0 14 7 0 0 0 0 5
0 7 0 0 0 0 15 6 1 0 0 10 0 12 8 0
8 0 0 13 16 0 0 11 0 0 15 0 0 0 4 7
0 9 0 0 0 0 7 0 0 11 0 16 14 0 0 0
0 11 3 0 15 12 0 0 8 0 0 0 0 10 0 0
1 0 4 0 0 0 3 0 0 2 0 6 16 0 0 15
```

Figura 3: Exemplo de um Sudoku$_{16}$.

3 Compilação e Execução

Digite os seguintes comandos em um terminal `Bash` a partir do diretório raiz do programa:

- **Compilação**: `make`
- **Execução (grade de** 9x9**)**: `./sudokun < ../inputs/input9.in`

4 Grupos, Avaliação e Entrega

O trabalho deverá ser realizado em grupos de **3 alunos**. Os alunos serão responsáveis por formar os grupos com auxilio da ferramenta "**Escolha de Grupos - Trabalho 1 (T1)**" disponível no Moodle. Os trabalhos serão apresentados nos dias definidos no cronograma disponível no Moodle.

Pelo menos um dos integrantes de cada grupo deverá submeter um arquivo ZIP contendo: (i) o código fonte em C da solução do trabalho; e (ii) um relatório em PDF (mínimo 2 páginas) explicando e justificando a solução adotada. A data/hora limite para o envio dos trabalhos é **27/09/2017 às 23h55min**. **Não será permitida a entrega de trabalhos fora desse prazo: trabalhos não entregues no prazo receberão nota zero.**

O professor irá avaliar não somente a **corretude** mas também o **desempenho** e a **clareza** da solução. Durante a apresentação, o professor irá avaliar o **conhecimento individual dos alunos sobre os conteúdos teóricos e práticos vistos em aula e sobre a solução adotada no trabalho**. A nota atribuída à cada aluno i no trabalho ($NotaTrabalho_i$) será calculada da seguinte forma, onde A_i é a nota referente à apresentação do aluno i e S é a nota atribuída à solução do trabalho:

$$NotaTrabalho_i = \frac{A_i * S}{10} \tag{1}$$

ATENÇÃO: como indicado pela fórmula mostrada acima, **a nota atribuída à solução adotada será ponderada pelo desempenho do aluno durante a apresentação do trabalho**. Por exemplo, se o professor atribuir nota 10 para a solução adotada pelo grupo mas o aluno receber nota 5 pela apresentação – devido ao desconhecimento dos conteúdos teóricos, práticos e/ou da solução do trabalho – a sua nota final do trabalho será 5. A ausência no dia da apresentação ou recusa de realização da apresentação do trabalho implicará em nota zero na apresentação, fazendo com que a nota atribuída ao aluno também seja zero.

BÔNUS: o grupo que entregar a solução com desempenho consideravelmente superior às demais soluções poderá receber $0,5$ pontos na Prova 1 (P1). Somente estarão aptos a receber esta bonificação os alunos que tiverem obtido nota igual ao superior a $8,0$ na apresentação.