Algoritmos Combinatórios 2013
Prof. Lucia Moura INE-UFSC

Lista de Exercícios #3 (100 pontos, peso 15%)
Entrega: 27 de junho as 8:20 (em aula)

Requerimentos para programas: Escreva seu programa em alguma linguagem de alto nível tais como C, C++, Java. Entregue pseudocódigo, programa e resultados de saída (note que se muitos testes são feitos, imprima apenas um exemplo de saída e resuma os resultados em tabelas). Por favor especificar a plataforma em que seus testes foram rodados (velocidade, RAM, sistema operacional).

1. (50 points) **SUDOKU via simulated annealing**

   Consulte a LE2 para introduo ao jogo de **SUDOKU**. Neste exercício voce vai implementar e testar um algoritmo de simulated annealing para resolver SUDOKU baseado no artigo: Rhyd Lewis, Metaheuristics can solve sudoku puzzles, *Journal of Heuristics* **13**, 2007.

   A entrada de seu algoritmo é um jogo de SUDOKU (um tabuleiro de Sudoku preenchido com os dados). Siga os seguintes requerimentos:

   - Solução inicial: preencha aleatoriamente cada uma das matrizes 3×3 de maneira a conter cada número em $\{1, 2, \ldots, 9\}$ exatamente uma vez. Jogadas nunca mudarão a frequência dos números em cada matriz 3×3.
   - Cada solução considerada na busca tem cada matriz 3×3 completa contendo cada um dos números em $\{1, 2, \ldots, 9\}$ exatamente uma vez. Linhas e colunas podem não satisfazer a propriedade desejada de ter cada número aparecendo exatamente uma vez.
   - Aqui especificamos uma função de custo a ser minimizada (substitua maximização de lucro por minimização de função de custo). Para cada linha i (coluna j) a contagem da linha rs_i (contagem de coluna cs_j, respectivamente) é o número de símbolos faltando na linha i (coluna j, respectivamente). O custo total de uma solução é a soma das contagens de suas linhas e colunas, isto é $(\sum_{i=1}^{9} rs_i) + (\sum_{j=1}^{9} cs_j)$. Obviamente, uma solução do jogo terá sido encontrada quando a função de custo for 0. Veja exemplos de cálculo de score na pagina 293 do artigo acima.
   - Cada vizinhp da solução actual consiste de uma solução obtida por troca de valores de duas células não fixas da mesma submatrix 3×3. A busca na vizinhança segue o paradigma de simulated annealing - então primeiro uma célula não fixa (célula que não é um dado) é selecionada ao acaso; depois um outra célula não fixa da mesma submatriz é escolhida ao acaso.

(a) Dê um pseudocódigo para o algoritmo de simulated annealing.

(b) Implemente seu método, experimentando para encontrar valores apropriados para a temperatura inicial t_0, fator de esfriamento α (o autor sugere $\alpha = 0.99$) e c_{max}. Note que o critério de parada deve ser quando o custo é 0 ou c_{max} iterações passaram, então c_{max} deve ser escolhido bem alto com a esperança de que nunca seja invocado.

(c) Teste seu método com os primeiros 500 dos milhares de Sudokus com 17 dados em:

`http://school.maths.uwa.edu.au/~gordon/sudoku17`

Escolha algumas combinações de parametros diferentes, e para cada uma delas dê estatísticas da solução de todos os jogos da coleção (tempo médio de execução, número médio de iterações, # de soluções bem sucedidas).

Nota: uma busca exaustiva feita por Gary McGuire determinou que 17 é o número mínimo de dados em um jogo de Sudoku com uma só solução, isto é qualquer confirguração de 16 dados sempre admite mais de uma solução.

2. (10 points) **Encontrar isomorfismos a mão.**
Exercício 7.1 do livro texto.

3. (10 points) **Certificado para árvores**
Faça o exercício 7.2 do livro texto. Simule o algorithm a mão, mostrando sua árvore e rótulos a cada passo.

4. (10 points) **Reverta o certificado de uma árvore**
Faça o exercicio 7.3 do livro texto. Mostre como a árvore via sendo construida passo a passo.

5. (20 points) **Certificado para grafos**
Simule a mão o Algoritmo 7.8, computando um certificado para o seguinte grafo. Mostre sua árvore de backtracking. (Note que antes de iniciar a primeira bifurcação, tem uma chamada para o procedimento REFINE).