Algoritmos Combinatórios 2013
Prof. Lucia Moura INE-UFSC

**Lista de Exercícios #2** (100 pontos, peso 15%)
Entrega: 13 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 pontos) **SUDOKU via backtracking**

   **SUDOKU** é um jogo com símbolos de 1 a 9 que são colocados em células de uma matriz $9 \times 9$ composta de nove $3 \times 3$ submatrizes, chamadas regiões. A matriz está parcialmente preenchida com alguns símbolos (os "dados"). A matriz tem que ser completa de modo que cada linha, coluna e região contenha exatamente uma occorrência de cada símbolo.

   Exemplo 1: fácil para humanos

   | | | | | | | | | |
   |---|---|---|---|---|---|---|---|---|
   | | 5 | | | | 1 | 4 | | |
   | 2 | | 3 | | | | 7 | | |
   | | 7 | | 3 | | | 1 | 8 | 2 |
   | | | 4 | | 5 | | | | 7 |
   | | | | 1 | | 3 | | | |
   | 8 | | | | 2 | | 6 | | |
   | 1 | 8 | 5 | | | 6 | | 9 | |
   | | | 2 | | | | 8 | | 3 |
   | | | 6 | 4 | | | | 7 | |

   Exemplo 2: médio para humanos

   | | | | | | | | | |
   |---|---|---|---|---|---|---|---|---|
   | | | 4 | | 5 | | | 6 | |
   | | 6 | | 1 | | | 8 | | 9 |
   | 3 | | | | | 7 | | | |
   | | 8 | | | | | 5 | | |
   | | | | 4 | | 3 | | | |
   | | | 6 | | | | | 7 | |
   | | | | 2 | | | | | 6 |
   | 1 | | 5 | | | 4 | | 3 | |
   | | 2 | | | 7 | | 1 | | |

   Exemplo 3: difícil para humanos

   | | | | | | | | | |
   |---|---|---|---|---|---|---|---|---|
   | 2 | | | 6 | 7 | | | | |
   | | | 6 | | | | 2 | | 1 |
   | 4 | | | | | | 8 | | |
   | 5 | | | | | 9 | 3 | | |
   | | 3 | | | | | | 5 | |
   | | | 2 | 8 | | | | | 7 |
   | | | 1 | | | | | | 4 |
   | 7 | | 8 | | | | 6 | | |
   | | | | | 5 | 3 | | | 8 |

   Neste exercício voc escreverá does algorithms de backtracking para resolver **SUDOKU**:

   (a) O primeiro algoritmo tentará completar a primeira posição disponível em ordem (por exemplo, da esquerda pra direita, de baixo pra cima).

   (b) O segundo algoritmo tentará completar a posição com o menor número de símbolos permitidos.

   Eficiência e clareza contam!

   Para cada um dos algoritmos:

   - Escreva um **pseudocódigo** para o algoritmo de backtracking que resolve o **SUDOKU**.
   - Implemente seu algoritmo e teste as instâncias dadas no web site do curso (se houver erros na entrada, especifique). A entrada para o seu programa consiste de uma matrix $9 \times 9$ representando a matriz de SUDOKU, once espaços em branco são representados por 0s.

     A saída dos seus programas deve consistir de:

     – matriz de entrada;
     – matriz de solução;
     – estatísticas sobre a performance do seu algoritmo tais como: número total de nós de backtracking, tempo de execução (tempo de CPU para a solução, não incluindo entrada e saida), etc.
   - Compare os resultados dos dois algoritmos baseando-se em uma tabela com as estatísticas de cada algoritmo nos mesmos dados de entrada. Discuta os resultados.

2. (50 pontos) **Backtracking para buscar códigos não lineares.**

   Se $x, y \in \{0,1\}^n$, lembre que $\textsc{Dist}(x, y)$ denota a distância de Hamming entre $x$ e $y$. Um código não-linear de comprimento $n$ e distância mínima $d$ é um subconjunto $\mathcal{C} \subseteq \{0,1\}^n$ tal que $\textsc{Dist}(x, y) \geq d$ para todo $x, y \in \mathcal{C}$. Denote por $A(n, d)$ o número máximo de $n$-tuplas em um código não-linear de comprimento $n$ e distância mínima $d$.

   - (35 pontos) Descreva um algoritmo de backtracking para computar $A(n, d)$ (dê pseudocódigo e qualquer outra explicação relevante).
     Implemente seu algoritmo e compute $A(n, 4)$ para $4 \leq n \leq 8$. Os valores corretos para $A(n, d)$ para valores pequenos de $n$ e $d$ podem ser encontrados na página:

     `http://www.win.tue.nl/~aeb/codes/binary-1.html`

     Para cada um dos seus testes, exiba os valores de entrada, resposta final e o número de nós da árvore de backtracking e tempo de CPU. Eficiência e clareza contam.
   - (15 pontos) Dê um pseudocódigo e um programa para o método de Knuth para estimar o tamanho da árvore de backtracking de seu algoritmo. Use este método para estimar o tamanho da árvore de backtracking para $4 \leq n \leq 11$. Para cada valor de $n$, escolha um número adequado $P$ de amostras e mostre a estimativa para pelo menos 5 valores diferentes de número de amostras igualmente espaçados no intervalo $[10, P]$.
     Esta estimativa aproxima bem o número de nós que você encontrou na parte anterior? (Caso negativo, aconselho checar a corretude das suas computações em ambas as partes).
   - Desafio de bônus (opcional, 10 pontos): use um algoritmo para determinar os valores de $A(9, 4)$ e $A(10, 4)$, que são 20 e 40 respectivamente; só trabalhe na questão bonus depois que todas as outras partes da lista foram completas.