Trabalho de Implementação 1

O "Problema das Oito Rainhas" é um problema combinatório clássico cuja resposta é uma configuração de um tabuleiro de xadrez com oito rainhas posicionadas de maneira que nenhuma possa atacar outra.
O problema é naturalmente generalizado para o "Problema das n Rainhas", cuja solução é um algoritmo que, dado o valor de n, apresenta uma configuração de um "tabuleiro de xadrez n×n" com n rainhas posicionadas de maneira que nenhuma possa atacar outra. É sabido que toda instância $n \notin \{2, 3\}$ admite resposta. Mais ainda, é possível computar uma resposta para toda instância $n \notin \{2, 3\}$ em tempo O(n).
Mesmo assim, o Problema das n Rainhas até hoje é usado como motivação para apresentar a técnica de "backtracking" por meio de um algoritmo como o abaixo, que recebe um "tabuleiro" parcialmente preenchido e devolve uma solução que "estende" este preenchimento a uma resposta, ou "não", caso isto não seja possível.

```
BacktrackRainhas(t)
      Se t é solução
          Devolva t
      l← primeira linha sem rainha em t
      Para cada c∈[1..n]
          Se uma rainha na posição (l, c) de t não ataca nenhuma das demais
              acrescente a t uma rainha na posição (l, c)
              r←BacktrackRainhas(t)
              Se r≠ "não"
                  Devolva r
      Devolva "não"
```

Outra solução para o problema consiste em considerar o grafo onde cada vértice é uma posição do tabuleiro, e dois vértices são vizinhos se rainhas posicionadas nas respectivas casas do tabuleiro podem atacar uma à outra. Neste caso, cada resposta para uma instância n do problema corresponde a um conjunto independente de tamanho n no grafo.
O Algoritmo ConjIndep(G, n, I, C) abaixo recebe um grafo G, um inteiro positivo n, um conjunto $I \subseteq V(G)$ independente em G e um conjunto $C \subseteq V(G) - I$ satisfazendo $\Gamma G(I) \cap C = \emptyset$, e devolve um conjunto R independente em G com n vértices satisfazendo $I \subseteq R \subseteq I \cup C$, ou "não" caso não exista tal conjunto.

```
ConjIndep(G, n, I, C)
      Se |I|=n
          Devolva I
      Se |I|+|C|<n
          Devolva "não"
      remova um vértice v de C
      R←ConjIndep(G, n, I∪{v}, C−ΓG(v))
      Se R≠ "não"
          Devolva R
      Devolva ConjIndep(G, n, I, C)
```

Este trabalho diz respeito à variante do "Problema das n Rainhas" onde algumas casas do tabuleiro são proibidas. Mais precisamente, dados um inteiro positivo n e um conjunto de posições C do "tabuleiro de xadrez n×n", queremos determinar uma configuração com o maior número possível de rainhas tal que as rainhas não sejam posicionadas em nenhuma das posições em C e de maneira que nenhuma ataque outra.

Formalmente o problema pode ser descrito assim:

n Rainhas com Casas Proibidas

Instância: Um inteiro positivo n e um conjunto $C \subseteq [1..n]\times[1..n]$

Resposta: Uma sequência $(c1, \ldots, cn) \in [0..n]n$ com o menor número de zeros possível, tal que rainhas dispostas nas posições (i, ci) para cada $1 \leq i \leq n$ com $ci \neq 0$ não podem se atacar nem ocupam posições em C (ci = 0 indica que nenhuma rainha é colocada na linha i).

Por exemplo, a instância do tabuleiro 8x8 onde as casas de ambas as diagonais são proibidas é a instância $(8, \{(i, j) \in [1..n]\times[1..n]: i = j \text{ ou } i + j = 9\})$; uma resposta desta instância é a sequência (2, 5, 7, 1, 3, 8, 6, 4).

O trabalho consiste em implementar duas soluções do problema das Rainhas com Casas Proibidas, uma baseada em "backtracking" e outra baseada na modelagem por meio de conjuntos independentes de um grafo, e compará-las quanto à eficiência.

A especificação do trabalho está em trabalho-1.tar.gz, onde você vai encontrar os seguintes arquivos.

trabalho-1/rainhas.h:
: a especificação do que deve ser implementado;

trabalho-1/rainhas.c:
: um esqueleto de implementação do especificado em rainhas.h;

trabalho-1/teste.c:
: um programa de teste a título de exemplo;

trabalho-1/makefile:
: um makefile com as opções de compilação que serão usadas na correção.

Entrega

O trabalho deve ser entregue sob a forma de um arquivo de nome fulano-sicrano.tar.gz, sendo que fulano e sicrano devem ser substituídos pelos login name dos autores.

O arquivo fulano-sicrano.tar.gz, uma vez expandido, deve conter (somente) os seguintes arquivos.

fulano-sicrano/rainhas.c:
: a implementação do especificado em trabalho-1/rainhas.h.

fulano-sicrano/readme.txt:
: texto comunicando tudo que seja relevante para a correção do trabalho.

O arquivo fulano-sicrano.tar.gz deve ser entregue como anexo de mensagem enviada para m.v.g.dasilva@gmail.com (Turma BCC1) ou renato.carmo.rc@gmail.com (Turma BCC2). O "Subject:" desta mensagem deve ser "Entrega do trabalho 1".

O prazo para a entrega é às 23h59min do dia 7 de julho.

Avaliação

Além dos habituais critérios de correção, aderência à especificação etc, sua implementação será avaliada também quanto ao de tempo de execução.

Todas as implementações serão submetidas a uma mesma bateria de instâncias. De acordo com o tempo de execução, as implementações serão classificadas e receberão uma nota proporcional à sua classificação. A nota nesta classificação corresponderá a 15% da nota total. Noutras palavras, 85 pontos da nota correspondem à correta implementação e os demais 15 pontos correspondem à eficiência da implementação, comparada com os demais trabalhos entregues.

Você é livre para aprimorar sua implementação usando quaisquer ideias, sejam elas heurísticas, sejam algoritmos e/ou estruturas de dados para melhorar o desempenho. Todas estas ideias devem ser explicadas no arquivo readme.txt.