%Validation\_nQueens\_Problem\_Solution

%{

Ce programme est destiné:

a) *pour vérifier l'exactitude de la solution Problème n-Queens pour un échiquier arbitraire de taille n x n,*

b) *vérifier l'exactitude d'une composition arbitraire de k reines réparties aléatoirement sur un échiquier de taille n x n.*

%}

%{

License: *Attribution-NonCommercial-ShareAlike CC BY-NC-SA*

“Cette licence permet à d'autres personnes de remixer, d'adapter et de s'appuyer sur votre travail de manière non commerciale, à condition qu'elles vous créditent et concèdent leurs nouvelles créations sous les mêmes conditions.”.

%}

%{

Auteur et développeur du projet - Grigoryan Eros (EricGrig), 2020

Je serai heureux si des sections du code, ou l'ensemble du programme dans son ensemble, seront utilisés à des fins scientifiques ou à des fins éducatives. En même temps, je vous serai reconnaissant si vous considérez qu'il est possible de faire référence à ma publication. C'est un élément de culture et un signe de respect mutuel.

Pour une utilisation commerciale d'une partie quelconque du code du programme, ou de l'ensemble du programme dans son ensemble, le consentement écrit de l'auteur est requis.

%}

%{

Cet algorithme est décrit en détail dans l'article: *Grigoryan E., Linear algorithm for solution n-Queens Completion problem,* [*https://arxiv.org/abs/1912.05935*](https://arxiv.org/abs/1912.05935) *.* Ce sera correct si vous lisez d'abord cette publication avant de commencer à analyser le code source. Cela rendra la description du programme plus transparente et réduira le nombre de questions possibles.

%}

%{

*Comment sont préparées les données initiales*?

Indiquez la taille du côté de l'échiquier par *n*.

Soit un tableau de taille annulé à une dimension *n*.

Si dans la *i-ème* rangée de l'échiquier la reine est placée en position *j*, alors, respectivement, dans la *i-ème* cellule du tableau de données, la valeur de *j* est écrite.

%}

%{

*Entrée de données*.

Nous écrirons les données d'entrée dans le tableau *Q*.

Le nom du fichier d'entrée *nQueens\_Test\_Solution.mat* doit être changé, si un autre fichier est sélectionné

%}

inputFileName= 'nQueens\_Test\_Solution.mat';

%inputFileName= 'kQueens\_Test\_Composition.mat';

iInfo=['Input file name: ' inputFileName];

disp(iInfo);

Q=importdata(inputFileName);

% Afficher la taille d'un échiquier

n=length(Q);

disp(' ');

tStr = sprintf(' The size of a chessboard = %d',n);

disp(tStr);

disp(' ')

% n2 – la taille des tableaux de contrôle diagonaux

n2=n\*2;

% Formation de tableaux de contrôle pour le contrôle diagonal.

D1=zeros(1,n2,'uint8');

D2=zeros(1,n2,'uint8');

B=zeros(1,n,'uint8');

%{

Nous déterminons le nombre de lignes sur l'échiquier dans lequel la reine ne se trouve pas, c'est-à-dire le nombre de lignes libres. Ce nombre est égal au nombre total de cellules nulles dans le tableau *Q*.

%}

nZero=sum(Q==0);

qError = 0;

%{ *qError* – indice de validation de décision. Si *qError == 0*, alors la solution est correcte, sinon, si *qError == 1*, alors la solution contient une erreur.

%}

tic

%{

Si *nZero == 0*, alors cela signifie que les données d'entrée sont une solution du problème n-Queens, sinon - les données d'entrée sont une composition.

%}

%{

L'essence de l'algorithme est que nous ne pouvons écrire qu'une seule fois 1 dans les cellules mises à zéro des tableaux de contrôle *D1, D2*, et *B*, sinon, l'une des trois conditions de la tâche sera violée.

%}

if nZero==0

for i=1:n

j=Q(i);

r=n+j-i;

t=j+i;

if D1(r)==0 && D2(t)==0 && B(j)==0

D1(r)=1;

D2(t)=1;

B(j)=1;

else

qError=1;

break

end

end

else

for i=1:n

if Q(i)>0

j=Q(i);

r=n+j-i;

t=j+i;

if D1(r)==0 && D2(t)==0 && B(j)==0

D1(r)=1;

D2(t)=1;

B(j)=1;

else

qError=1;

break

end

end

end

end

toc

%{

*k* – le nombre d'éléments différents de zéro dans le tableau *Q* (taille de la composition)

%}

if qError==0

if nZero==0

disp('Solutions is ok!')

else

tStr = sprintf('Composition size = %d', n-nZero);

disp(tStr);

disp('Composition is ok!')

end

else

tStr = sprintf('Error in solution in row = %d',i);

disp(tStr)

end