جامعة دمشق – كلية الهندسة المعلوماتية

مشروع مبادئ الذكاء الاصطناعي

Nurikabe Puzzle

إعداد الطلاب :

خضر عمار يوسف

ثراء خالد غزالي

ماريال حسن شاهيني

مايا باسل عبد لله

سارة أنور الجرماني

سنا نزيه عيد

المهندسة المشرفة : عبير الكجك

المقدمة :

لعبة نوريكابي من ألعاب الأحاجي التي يلعبها العب واحد والتي تعتمد على المنطق، سنقوم بكتابة برنامج يقوم بحل هذه اللعبة

باستخدام لغة البرمجة برولوغ.

تتألف اللعبة من جدول مقسم إلى خلايا ، يمكن أن يكون للجدول قياسات مختلفة. بعض هذه الخلايا يحوي على أرقام،

هدف اللعبة هو ملء خلايا الجدول الغير حاوية على أرقام إما بخلية (زرقاء) تمثل بحر (أو خلية خضراء) تمثل (أرض) , يحوي الجدول على جزر بعدد الخلايا المرقمة، بحيث تكون الخلية المرقمة هي إحدى خلايا الجزيرة التي عدد خلاياها هو الرقم داخل الخلية.

يتم ملء الخلايا وفق القواعد التالية :

- لا يمكن تعديل لون الخلية الحاوية على رقم , و لا تحتوي الرقعة على جزر خلاياها ليســـت إحدى الخلايا المرقمة.
- كـل جزيرة معزولـة عن الجزر الأخرى , اي أن خلايـاهـا لا تلتقي عمودياً أو أفقيـاً , ولكن من الممكن أن تتلامس بشكل قطري.
- يجب على الخلايا الزرقاء أن تكون متصلة , أي أنه لا يمكن أن تشكل بحيرة معزولة ضمن أرض.
- لا يمكن وصل الخلايا الزرقاء لتشكل كتل من الحجم 2x2 , بينما يمكن لخلايا الجزر أن تحتوي هكذا كتل.

أولاً : تحقيق المسألة بهدف التحقق من صحة الحل :

تمثيل المسألة :

1 - نمثل الخلايا الثابتة ضـمن اللوح عن طريق اسـناديات، بحيث تعبر عن علاقة بين ثلاث معاملات : السـطر و العمود و قيمة الرقم فيها، بحيث يكون لدينا حقيقة من أجل كل خلية ثابتة ضمن اللوح الخاص بالعبة (Grid).

```
fxd_cell(Row,Col,Num).
```

من أجل المثال المعطى بملف التقرير :

```
fxd_cell(1,2,3).

fxd_cell(1,6,1).

fxd_cell(3,1,2).

fxd_cell(3,4,1).

fxd_cell(5,2,1).

fxd_cell(5,5,2).

fxd_cell(6,3,2).

fxd_cell(7,1,1).

fxd_cell(7,5,1).

fxd_cell(7,7,6).
```

2 - نمثل خلايا الحل بإسناديات (علاقة بين سطر و عمود و لون) مولدة بشكل ديناميكي خلال الحل.

```
solve_cell(Row,Col,Color).
```

من أجل المثال المعطى بملف التقرير :

```
solve_cell(1, 1, blue).
solve_cell(1, 2, green).
solve_cell(1, 3, green).
solve_cell(1, 4, green).
solve_cell(1, 5, blue).
solve_cell(1, 6, green).
solve_cell(1, 7, blue).
solve_cell(2, 1, blue).
solve_cell(2, 2, blue).
solve_cell(2, 3, blue).
```

```
solve_cell(2, 4, blue).
solve_cell(2, 5, blue).
solve_cell(2, 6, blue).
solve_cell(2, 7, blue).
solve_cell(3, 1, green).
solve_cell(3, 2, green).
solve_cell(3, 3, blue).
solve_cell(3, 4, green).
solve_cell(3, 5, blue).
solve_cell(3, 6, green).
```

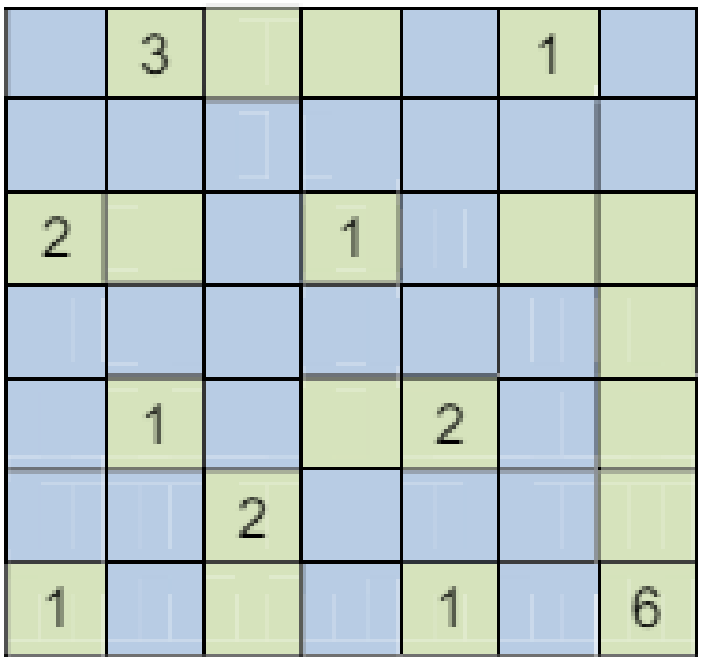
```
solve_cell(3, 7, green).
solve_cell(4, 1, blue).
solve_cell(4, 2, blue).
solve_cell(4, 3, blue).
solve_cell(4, 4, blue).
solve_cell(4, 5, blue).
solve_cell(4, 6, blue).
solve_cell(4, 7, green).
solve_cell(5, 1, blue).
solve_cell(5, 2, green).
```

```
solve_cell(5, 3, blue).
solve_cell(6, 6,blue).
solve_cell(6, 7,green).
solve_cell(7, 1,green).
solve_cell(7, 2,blue).
solve_cell(7, 3,green).
solve_cell(7, 4,blue).
solve_cell(7, 5,green).
solve_cell(7, 6,blue).
solve_cell(7, 7,green).
```

```
solve_cell(6, 2,blue).
solve_cell(6, 3,green).
solve_cell(6, 4,blue).
solve_cell(6, 5,blue).
```

```
solve_cell(5, 4, green).
solve_cell(5, 5, green).
solve_cell(5, 6, blue).
solve_cell(5, 7, green).
```

و هو ما تمثله هذه ال Grid :



إجرائية التحقق من صحة الحل (Solved) :

```
solved:-
    no_2x2_sea,
    one_sea,
    one_fixed_cell_in_island,
    island_number_equals_size.
```

و هي تتكون من قواعد جزئية :

1 – no_2x2_sea : تتحقق هذه الاســـنادية من عدم وجود كتل من الخلايا الزرقاء بحجم 2x2 أو أكبر , و هي معرفة وفق :

```
% ? - - -
```

2 – one_sea : تتحقق هذه الاسنادية من ان جميع الخلايا الزرقاء متصلة مع بعضها البعض (تشكّل بحر واحد) ,

و هي معرفة وفق :

```
% ? - - -
```

معينة (X,Y) و تستدعي الاسنادية sea((X,Y),Sea) على هذه الخلية ثم تحسب طول السلسلة الراجعة Sea و تقارنه مع عدد الخلايا الزرقاء الكلي ,فإذا كانا متساويان هذا يعني أن جميع الخلايا الزرقاء متصلة مع بعضها البعض , و ان لم يكونا متساويان فإن الخلايا الزرقاء ليست جميعها متصلة مع بعضها البعض (البحر ليس وحيداً).

3 –one_fixed_cell_in_island : تتحقق هذه الاسنادية من أن كل جزيرة تحتوي على خلية ثابتة واحدة و واحدة فقط , و هي معرفة وفق :

```
% ? - - -
```

```
% ? - - -
```

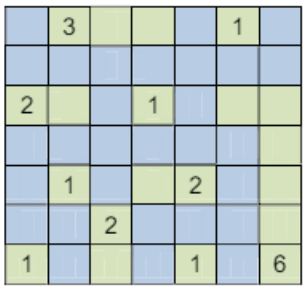
للاسنادية islands(LandCells,Islands) و هي التي تقوم بالمذكور سابقاً حقاً, بالنسبة للاسنادية get_island_fixed_cells(Island,IslandFixedCells) فهي تقوم بإيجاد الخلايا الثابتة ضمن الجزيرة الممررة لها على شكل List و هذه ال List هي IslandFixedCells , بالنسبة للاسنادية get_island_fixed_cells_helper([(X,Y)|Rest],Acc,FixedCells) فهي الاسنادية المساعدة للاسنادية get_island_fixed_cells(Island,IslandFixedCells) و هي التي تقوم بالمذكور سابقاً فعلياً , و بالنسبة للاسنادية check_islands_each_has_exactly_one_fixed_cell([Island|Islands]) فهي تقوم و بالاعتماد على الاسناديات المذكورة سابقاً بالمرور على كل جزيرة من الجزر و ايجاد الخلايا الثابتة لكل جزيرة منها ثم مقارنة عدد الخلايا الثابتة لكل جزير مع ال 1 , و أخيراً تقوم الاسنادية التي تمثل القاعدة بجمع خلايا الجزر(الخلايا الخضراء) و تمريرها للاسنادية check_islands_each_has_exactly_one_fixed_cell([Island|Islands]) و التي تتحقق من المطلوب.

4 – island_number_equals_size : تتحقق هذه الاسنادية من أن عدد الخلايا بكل جزيرة يساوي العدد الموجود على الخلية الثابتة التابعة للجزيرة , و هي معرفة وفق :

```
% ? - - -
```

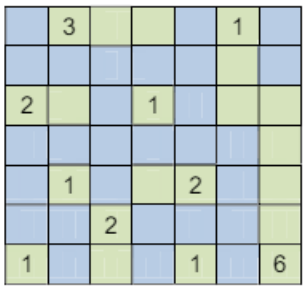
بالاستفادة من الاسناديات المستعملة في القاعدة السابقة , تقوم الاسنادية check_islands_each_has_size_equals_fixed_cell_number([Island|Islands]) بالمرور على كل جزيرة ثم ايجاد الخلايا الثابتة لكل جزيرة و عددها واحد لكل جزيرة حسب القاعدة السابقة , ثم مقارنة حجم كل جزيرة (عدد خلاياها) مع العدد الموجود (المرفق) مع الخلية الثابتة التابعة لهذه الجزيرة , 1 , و أخيراً تقوم الاسنادية التي تمثل القاعدة بجمع خلايا الجزر(الخلايا الخضراء) و تمريرها للاسنادية check_islands_each_has_size_equals_fixed_cell_number([Island|Islands]) و التي تتحقق من المطلوب.

و بهذا عند استدعاء الاسنادية solved نتحقق من الحل الموجود لدينا و المعرف على شكل حقائق , نتيجة التنفيذ لبعض الحالات :



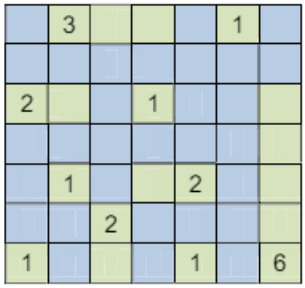

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- solved.
true
```

Violate one_sea



Violate one_fixed_cell_in_island

Violate no_2x2_sea



Violate island_number_equal_size

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- solved.
false.
```

ثانياً : حل المسألة باستخدام القواعد :

1 – تهيئة اللعبة :

```
:-dynamic solve_cell/3.
:-dynamic fxd_cell/3.
% Predicate to initialize the grid
initialize_grid(Rows, Cols, FixedCells) :-
    retractall(solve_cell(_, _, _)),
    retractall(fxd_cell(_, _, _)),
    create_grid(Rows, Cols),
    initialize_fixed_cells(FixedCells).
% Helper predicate to create the grid
create_grid(Rows, Cols) :-
    between(1, Rows, Row),
    between(1, Cols, Col),
    assertz(solve_cell(Row, Col, empty)),
    fail.
create_grid(_, _).
% Predicate to set fixed cells directly from a list
initialize_fixed_cells([]).
initialize_fixed_cells([(Row, Col, Value)|Rest]) :-
    assertz(fxd_cell(Row, Col, Value)),
    retract(solve_cell(Row,Col,empty)),
    assertz(solve_cell(Row, Col, green)),
    initialize_fixed_cells(Rest).
```

نســتدعي الاســنادية initialize_grid(Rows,Cols,FixedCells) عندما نريد أن نقوم بتهيئة الرقعة (Grid) , حيث تقوم هذه الاسنادية بحذف كل الاسناديات الديناميكية و المرتبطة بحلول سابقة , ثم تهيئة رقعة جديدة باستدعاء الاســنادية create_grid(Rows,Cols) و التي تقوم على اســاس جعل جميع الخلايا باســتثناء الخلايا الثابتة تحمل القيمة الابتدائية empty و نكتب solve_cell(X,Y,empty) , اما بالنســبة للخلاية الثابتة فيتم تهيئتها على انها خلايا ثابتة و خلايا جزر في نفس الوقت , أي : solve_cell(X,Y,green) و fxd_cell(X,Y,N) , وذلك من خلال الاســنادية initialize_fixed_cells([(Row,Col,Value)|Rest]) .

2 – طباعة الرقعة : مجرد مرور على الخلايا و طباعة لونها اذا لم تكن خلية ثابتة , اما اذا كانت خلية ثابتة اطبع قيمتها (رقمها).

```
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
print_grid:-
    grid(R,C),\+print_grid_helper(R,C).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
print_grid_helper(Rows, Cols) :-
    between(1, Rows, Row),
    between(1, Cols, Col),
    (fxd_cell(Row, Col, Value) -> write(Value) ; (solve_cell(Row, Col, State) -> print_cell(State) ; write(' '))),
    (Col =:= Cols -> nl; true),
    fail.
print_grid(_, _).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
print_cell(green) :- write('G').
print_cell(blue) :- write('B').
print_cell(empty) :- write('.').
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

أمثلة على طباعة الرقعة :

المثال المرفق بنص إعلان المشروع :

```
.3...1.
.......
2..1...
.......
.1..2..
..2....
1...1.6
```

قبل الحل (بعد التهيئة)

```
B3GGB1B
BBBBBBB
2GB1BGG
BBBBBBG
B1BG2BG
BB2BBBG
1BGB1B6
```

بعد الحل

3 – حل اللعبة :

```
solve_logically:-
    solve_for_fixed_cells_with_clue_equals_one,
    solve_for_cells_with_all_neighbors_blue,
    seas_one_way_out,
    solve_for_fixed_cells_that_are_one_cell_separated,
    isolate_completed_islands,
    solve_for_diag_adj_fxd_cells,
    solve_for_no_2x2_blue_blocks,
    solve_for_unreachable_cells.
solved:-
    no_2x2_sea,
    one_sea,
    island_number_equals_size,
    one_fixed_cell_in_island.
solve_logic(N):-
    findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
    length(EmptyCells,L),
    L =\= N,
    solve_logically,
    solve_logic(L).
solve_logic(N):-
    findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
    length(EmptyCells,L),
    L =:= N,
    solve_logically.
solve_nurikabe:-
    solve_logic(0),
    solve_backtracking.
```

```
solve_backtracking :-
    % Get all empty cells
    findall((Row, Col), solve_cell(Row, Col, empty), EmptyCells),
    % Try to solve the puzzle
    solve_cells(EmptyCells).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
solve_cells([]) :-
    solved , nl , print_grid , nl , writeln("Nurikabe Solved :)").
% Recursive case: try to solve for one cell and continue
solve_cells([(Row, Col)|Rest]) :-
    % Try to color the cell blue
    color_cell_blue(Row, Col),
    solve_cells(Rest);
    % If it fails, backtrack and try to color the cell green
    color_cell_green(Row, Col),
    solve_cells(Rest).
```

تقوم الاسنادية solve_nurikabe بحل اللعبة و ذلك على مرحلتين :

1 – الحل باستخدام مجموعة من الاستراتيجيات المنطقية و الممثلة بالاسنادية solve_logically و المقسومة الى عدة اسناديات كل منها يمثل استراتيجية حل منطقية (سيتم شرح و عرض كود كل استراتيجية بعد هذه الفقرة).

2 – الحل باستخدام ال Backtracking , وذلك عن طريق الاسنادية solve_backtracking و التي تقوم بتجريب كل الحلول الممكنة بتلوين كل خلية مرة باللون الاخضر و مرة باللون الازرق, و بعد الانتهاء من التجريب يتحقق من صحة الحل و في حال الصحة (true) يطبع ال Grid و يطبع الرسالة "Nurikabe Solved 😊" , و في حال عدم قدرته على الحل يعيد False .

- تقوم الاسنادية solve_nurikabe بالاستعانة بالاسنادية solve_logic(N) و التي تقوم بالحل بنائاً على الاسنادية solve_logically , و تقوم باستدعاء هذه الاسنادية عدة مرات إلى أن تصل الى مرحلة لا يمكن أن تحل بهذه الاسنادية solve_logically (الاسترتيجيات المنطقية) اي خلية أخرى عندها تتوقف الاسنادية solve_logic عن العمل , ويأتي دور الاسنادية solve_backtracking لتجرب كل الحلول الممكنة على ما تبقى من الخلايا التي عجزت solve_logically عن حلها.

```
% ? - - -
```

```
% ? I - Solving For Fixed Cells With Clue Equals One :
solve_for_fixed_cells_with_clue_equals_one:-
    findall((X,Y),fxd_cell(X,Y,1),FixedCellsWithClueEqualsOne),
    color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue(FixedCellsWithClueEqualsOne).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
color_list_blue([]).
color_list_blue([(X,Y)|Rest]):-
    color_cell_blue(X,Y) , print_grid , nl , nl , color_list_blue(Rest).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue([]).
color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue([(X,Y)|Rest]):-
    findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
    color_list_blue(Neighbors),color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue(Rest).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

تقوم الاســـنادية color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue([(X,Y)|Rest]) بالمرور على جميع الخلايا الثابتة ذات الدليل 1 و تلوين جيرانها الأفقية و الشـــاقولية باللون الأزرق , الاســـنادية color_list_blue([(X,Y)|Rest]) تقوم بتلوين List الخلايا الممررة لها باللون الأزرق , واســـنادية الاســـتراتيجية solve_for_fixed_cells_with_clue_equals_one تقوم بتجميع الخلايا الثابتة ذات الدليل واحد وتمريرها للاسنادية color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue([(X,Y)|Rest]).

2 – تلوين الخلايا المحاطة من جميع الاتجاهات بخلايا زرقاء باللون الأزرق :

و هي نتيجة حتمية للمحافظة على استمرارية الماء (الخلايا الزرقاء) :

```
% ? II - Solving For Cells With All Blue Neighbors:
solve_for_cells_with_all_neighbors_blue:-
    findall((X,Y),solve_cell(X,Y,empty),CellsWithAllNeighborsBlue),
    color_the_cells_that_all_their_neighbors_are_blue(CellsWithAllNeighborsBlue).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
check_for_neighbors_being_blue([]).
check_for_neighbors_being_blue([(X,Y)|Neighbors]):-
    solve_cell(X,Y,blue) , check_for_neighbors_being_blue(Neighbors).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
color_the_cells_that_all_their_neighbors_are_blue([]).
color_the_cells_that_all_their_neighbors_are_blue([(X,Y)|Rest]):-
    findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
    \+check_for_neighbors_being_blue(Neighbors) , color_the_cells_that_all_their_neighbors_are_blue(Rest).
color_the_cells_that_all_their_neighbors_are_blue([(X,Y)|Rest]):-
    findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
    check_for_neighbors_being_blue(Neighbors) , color_cell_blue(X,Y), print_grid , nl , nl , color_the_cells_that_all_their_neighbors_are_blue(Rest).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

تقوم الاســـنادية (check_for_neighbors_being_blue([(X,Y)|Neighbors]) بالتحقق من كون الخلايا المجاورة (الأفـــقـــيـــة و الشـــــاقـــولـــيـــة) زرقـــاء الـــلـــون , و بـــالـــنســـــبـــة لـــلاســـــنـــاديـــة (color_the_cells_that_all_their_neighbors_are_blue([(X,Y)|Rest]) فهي تقوم بـالمرور على جميع الخلايـا الفارغة (solve_cell(X,Y,empty , و التحقق من كون جيرانها (الأفقية و الشـــاقولية) زرقاء اللون , و ذلك اعتماداً على الاســـنادية (check_for_neighbors_being_blue([(X,Y)|Neighbors]) , ثم تقوم بتلوين هذه الخلية باللون الأزرق , و ذلك في حال كانت نتيجة الاســـنادية (check_for_neighbors_being_blue([(X,Y)|Neighbors]) هي true , و في حال كانت نتيجتها false فهي تترك الخلية فارغة كما هي , وأخيراً الاســنادية التي تمثل الاســتراتيجية فـــهـــي تـــمـــرر List الـــخـــلايـــا ال empty لـــلاســـــنـــاديـــة (color_the_cells_that_all_their_neighbors_are_blue([(X,Y)|Rest]).

3 – تلوين الخلايا التي لها مجاور واحد (أفقي أو شاقولي) فارغ , وبقية المجاورات (الأفقية أو الشاقولية) ليس أي منها باللون الأزرق , باللون الأزرق :

و ذلك للمحافظة على استمرارية البحر (الخلايا الزرقاء) :

```
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% ? III - Solving For Sea Cells With One Way Out :
seas_one_way_out:-
    findall((X,Y),solve_cell(X,Y,blue),SeaCells),
    seas_one_way_out_helper(SeaCells).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
seas_one_way_out_helper([]).
seas_one_way_out_helper([(X,Y)|Rest]):-
    sea_one_way_out(X,Y) ,seas_one_way_out_helper(Rest).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
sea_one_way_out(X,Y):-
    grid(R,C) , Xr is X + 1 ,  Xl is X-1  , Yu is Y-1 ,  Yd is Y+1 ,
    (
        (
            Xl > 0,
            solve_cell(Xl,Y,empty),
            (solve_cell(Xr,Y,green);Xr > C),
            (solve_cell(X,Yu,green);Yu =< 0),
            (solve_cell(X,Yd,green);Yd > R),
            color_cell_blue(Xl,Y)
        )
        ;
        (
            Xr =< C ,
            (solve_cell(Xl,Y,green);Xl =< 0),
            solve_cell(Xr,Y,empty),
            (solve_cell(X,Yu,green);Yu =< 0),
            (solve_cell(X,Yd,green);Yd > R),
            color_cell_blue(Xr,Y)
        )
        ;
        (
            Yu > 0 ,
            (solve_cell(Xl,Y,green);Xl =< 0),
            (solve_cell(Xr,Y,green);Xr > C),
            solve_cell(X,Yu,empty),
            (solve_cell(X,Yd,green);Yd > R),
            color_cell_blue(X,Yu)
        )
        ;
        (
            Yd =< R ,
            (solve_cell(Xl,Y,green);Xl =< 0),
            (solve_cell(Xr,Y,green);Xr > C),
            (solve_cell(X,Yu,green);Yu =< 0),
            solve_cell(X,Yd,empty),
            color_cell_blue(X,Yd)
        )
    ),print_grid , nl , nl.
sea_one_way_out(_,_).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

تتحقق الاســـنادية sea_one_way_out(X,Y) من كون الخلية (X,Y) خلية زرقاء , و لها مجاور واحد فارغ و بقية المجـاورات ليس أي منهـا خليـة زرقـاء , ثم تقوم بتلوين ذلـك المجـاور الفـارغ بـاللون الأزرق , الاســـنـادية seas_one_way_out([(X,Y)|Rest]) , تقـوم بـالمرور على جميـع الخلايـا الزرقـاء , وتطبـق الاســـنـادية sea_one_way_out(X,Y) عليها , و أخيراً الاسـنادية التي تمثل الاسـتراتيجية تمرر List الخلايا الزرقاء للاسـنادية seas_one_way_out([(X,Y)|Rest]) .

4 – تلوين الخلايا المجاورة (أفقياً و شاقولياً) لخلايا الجزر المكتملة باللون الأزرق :

و ذلك لأن الجزر لا يمكن أن تلتقي أفقياً أو شاقولياً :

```
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% ? IV - Solving For Complete Islands :
isolate_completed_islands:-
    findall((X,Y,N),fxd_cell(X,Y,N),FixedCells),
    isolate_completed_islands_helper(FixedCells).
isolate_completed_islands_helper([]).
isolate_completed_islands_helper([(X,Y,N)|FixedCells]):-
    island((X,Y),Island),
    length(Island,L),
    L =:= N,
    surround_the_completed_island_with_blue(Island),
    isolate_completed_islands_helper(FixedCells).
isolate_completed_islands_helper([(X,Y,N)|FixedCells]):-
    island((X,Y),Island),
    length(Island,L),
    L =\= N,
    isolate_completed_islands_helper(FixedCells).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
surround_the_completed_island_with_blue([]).
surround_the_completed_island_with_blue([(X,Y)|Island]):-
    findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
    surround_the_completed_island_cell_with_blue_helper(Neighbors),
    surround_the_completed_island_with_blue(Island).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
surround_the_completed_island_cell_with_blue_helper([]).
surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]):-
    solve_cell(X,Y,empty) , color_cell_blue(X,Y) , print_grid , nl , nl , surround_the_completed_island_cell_with_blue_helper(Neighbors).
surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]):-
    \+solve_cell(X,Y,empty) , surround_the_completed_island_cell_with_blue_helper(Neighbors).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

حيـث تقوم الاســـنـادية surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]) بالمرور على الخلايا المجاورة الأفقية و الشـاقولية لكل خلية من خلايا الجزيرة و تلوينها باللون الأزرق في حال كانت فارغة , اما بالنســـبة للاســـنادية surround_the_completed_island_cell_with_blue ([(X,Y)|Island]) فهي تقـوم بـالمرور على كـل خليـة من خلايـا الجزيرة , ثم إيجـاد مجـاوراتهـا و تمريرهـا للاســـنـادية surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]) , و بالنســـبة للاســـنادية isolate_completed_islands_helper([(X,Y,N)|FixedCells]) فهي تمر على كـل الخلايـا الثـابتـة , و تتحقق من كون الجزيرة الخاصة بهذه الخلية الثابتة مكتملة , وذلك بمقارنة عدد خلايا الجزيرة بالدليل المرفق بالخلية الثابتة ,

و في الحال تساويا فإنها تمرر الجزيرة للاسنادية surround_the_completed_island_cell_with_blue ([(X,Y)|Island]) ليتم تحويطها بالماء , وفي حال لم يتساويا تترك الجزيرة على الحالها و تنتقل للجزيرة التالية , و أخيراً الاسنادية التي تمثل الاستراتيجة فهي تمرر List الخلايا الثابتة للاسنادية isolate_completed_islands_helper([(X,Y,N)|FixedCells]) .

5 – تلوين الخلايا المجاورة لخليتين ثابتتين متجاورتين قطرياً باللون الأزرق :

و ذلك لأن الجزر لا يمكن أن تلتقي أفقياً أو شاقولياً :

Must Be Water	Fixed Cell
Fixed Cell	Must Be Water

Fixed Cell	Must Be Water
Must Be Water	Fixed Cell

```
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% ? V - Solving For Diagonally Adjacent Fixed Cells :
solve_for_diag_adj_fxd_cells:-
    findall((X,Y),fxd_cell(X,Y,_),FixedCells),
    solve_for_diag_adj_fxd_cells_helper(FixedCells).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
solve_for_diag_adj_fxd_cells_helper([]).
solve_for_diag_adj_fxd_cells_helper([(X,Y)|FixedCells]):-
    Xur is X+1,
    Yur is Y+1,
    fxd_cell(Xur,Yur,_),
    color_cell_blue(Xur,Y),
    color_cell_blue(X,Yur), print_grid , nl , nl , solve_for_diag_adj_fxd_cells_helper(FixedCells).
solve_for_diag_adj_fxd_cells_helper([(X,Y)|FixedCells]):-
    Xur is X+1,
    Yur is Y+1,
    \+fxd_cell(Xur,Yur,_),
    solve_for_diag_adj_fxd_cells_helper(FixedCells).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

حيث تقوم الاســـنادية solve_for_diag_adj_fxd_cells_helper([(X,Y)|FixedCells]) بالمرور على جميع الخلايا الثابتة و التحقق من كون أن مجاورها القطري هو خلية ثابتة , وفي حال الاثبات فإنها تلون الخلايا المجاورة لهاتين الخليتين باللون الأزرق كما وضّــحنا بالشــكل أعلاه , وفي حال النفي فهي لاتقوم بأي تغيير و تنتقل للخلايا الثابتة الأخرى , و بالنسبة للاسنادية التي تمثل الاستراتيجية فهي تمرر List الخلايا الثابتة للاسنادية المساعدة.

6 – تلوين الخلايا التي لو لونت باللون الأزرق كانت لتشكل كتل ماء 2x2 ; باللون الأخضر :

لمنع تشكل كتل الخلايا الزرقاء بحجم 2x2 أو أكبر :

```
% ? - - -
```

حيث تقوم الاســـنادية if_it_will_form_2x2_blue_block_color_it_green(X,Y) بالتحقق من أنه لو لونت الخلية (X,Y) باللون الأزرق لشكلت كتلة زرقاء بحجم 2x2 أم لا , وفي حال الاثبات تلون الخلية باللون الاخضر , وفي حال النفي تترك الخلية فارغة على حالها , و بالنســبة للاســنادية prevent_2x2_blue_blocks([(X,Y)|Rest]) فـهـي تـمـر عـلـى جـمـيـع الـخـلايـا الـفـارغـة و تـطـبـق عـلـيـهـا الاســـنـادية if_it_will_form_2x2_blue_block_color_it_green(X,Y) , و أخيراً الاســـنادية التي تمثل الاســـتراتيجية فهي تمرر List الخلايا الفارغة للاسنادية prevent_2x2_blue_blocks([(X,Y)|Rest]).

7 – تلوين الخلايا الواقعة مباشرةً بين خليتين ثابتتين أفقياً أو شاقولياً باللون الأزرق :

و ذلك لأن الجزر لا يمكن أن تلتقي أفقياً أو شاقولياً :

```
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% ? VII - solving for fixed cells that are vertically or horizontally separated by one cell :
solve_for_fixed_cells_that_are_one_cell_separated:-
    findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
    separate_fxd_cells(EmptyCells).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
fxd_cell_separator(X,Y):-
    Xr is X + 1,
    Xl is X - 1,
    Yu is Y - 1,
    Yd is Y + 1,
    (
        (fxd_cell(Xr,Y,_),fxd_cell(Xl,Y,_))
        ;
        (fxd_cell(X,Yd,_),fxd_cell(X,Yu,_))
    ),color_cell_blue(X,Y), print_grid , nl , nl.
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
separate_fxd_cells([]).
separate_fxd_cells([(X,Y)|Rest]):-
    fxd_cell_separator(X,Y) , separate_fxd_cells(Rest).
separate_fxd_cells([(X,Y)|Rest]):-
    \+fxd_cell_separator(X,Y) , separate_fxd_cells(Rest).
% ? - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

حيث تقوم الاسـنادية fxd_cell_separator(X,Y) بالتحقق من كون الخلية (X,Y) واقعة مباشـرةً أفقياً أو شـاقولياً بين خليتين ثابتتين ,ففي حال الاثبات يتم تلوين الخلية باللون الأزرق و في حال النفي تبقى الخلية فارغة على حالها, اما الاســـنادية separate_fxxd_cells([(X,Y)|Rest]) فهي تمر على جميع الخلايا الفارغة وتطبق عليها الاسـنادية fxd_cell_separator(X,Y), و أخيراً الاسـنادية التي تمثل الاسـتراتيجية فهي تمرر List الخلايا الفارغة للاسنادية separate_fxxd_cells([(X,Y)|Rest]) .

8 – تلوين الخلايا التي لايمكن الوصول إليها من أي خلية ثابتة باللون الأزرق :

```
% ? - - -
```

حيث تقوم الاسنادية find_path((X,Y),C,Vis,Path) بإيجاد مسار من الخلايا طوله C و ينطلق من الخلية (X,Y) , اما الاسنادية find_paths((X,Y),L ,Path) فهي تقوم و بالاعتماد على الاسنادية السابقة بإيجاد جميع المسارات الممكنة التي طولها L و تبدأ من الخلية (X,Y) , بالنسبة للاسنادية find_reachable_cells([(X,Y,N)|Rest],Reachable Cells ,Acc) فهي تقوم بإيجاد كل المسارات الممكنة لكل خلية ثابتة و التي طولها الدليل المرفق بالخلية الثابتة , و تعيد List بهذه المسارات كلها , بالنسبة للاسنادية color_unreachable_cells_by_blue([(X,Y)|Rest]) فهي تمرر List الخلايا الثابتة للاسنادية find_reachable_cells([(X,Y,N)|Rest],Reachable Cells ,Acc) و بعدها تقوم بعمل flatten لل List الراجعة من هذه الاسنادية والتي تمثل List من المسارات (List Of Lists) وهي ال NestedReachableCells, فنحصل على List من الخلايا و التي تمثل الخلايا التي يمكن الوصول اليها من الخلايا الثابتة لل Grid الخاصة باللعبة وهذه ال List هي ReachableCells ثم تمر هذه الاسنادية على كل الخلايا الفارغة و تتحقق فيما اذا كانت الخلية تنتمي لل ReachableCells ففي حال الإيجاب تضل الخلية فارغة على حالها , و في حال النفي يتم تلوين الخلية باللون الأزرق ,و أخيراً الاسنادية التي تمثل الاستراتيجية فهي تمرر List الخلايا الفارغة للاسنادية color_unreachable_cells_by_blue([(X,Y)|Rest]) .

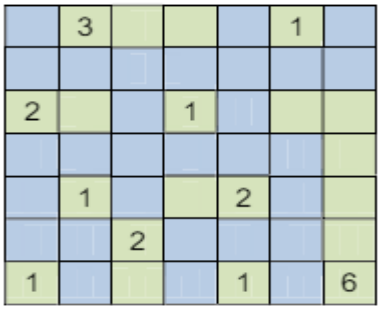
- حل بعض الأمثلة :

1 – المثال الأول (المثال المرفق بنص الإعلان) :

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- initialize_grid(7,7,[(1,2,3),(1,6,1),(3,1,2),(3,4,1),(5,2,1),(5,5,2),(6,3,2),(7,1,1),(7,5,1),(7,7,6)]).
true.
```

```
2 ?- solve_nurikabe.
```

```
B3GGB1B
BBBBBBB
2GB1BGG
BBBBBBG
B1BG2BG
BB2BBBG
1BGB1B6

Nurikabe Solved :)
true .
```



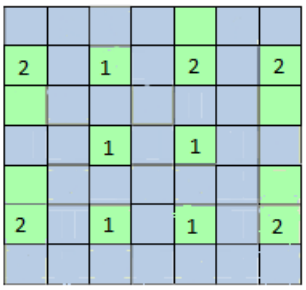
و هو الحل الفعلي لهذه ال Grid , طبعاً البرنامج يطبع خطوات الحل لكننا عرضنا النتيجة النهائية فقط.

2 – المثال الثاني :

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- initialize_grid(7,7,[(2,1,2),(2,3,1),(2,5,2),(2,7,2),(4,3,1),(4,5,1),(6,1,2),(6,3,1),(6,5,1),(6,7,2)]).
true.
```

```
2 ?- solve_nurikabe.
```

```
BBBBGBB
2B1B2B2
GBBBBBG
BB1B1BB
GBBBBBG
2B1B1B2
BBBBBBB

Nurikabe Solved :)
true .
```



و هو الحل الفعلي لهذه ال Grid , تم جريب ثلاث أمثلة أخرى ضمن الكود.

Thank

You