



جامعة دمشق - كلية الهندسة المعلوماتية

مشروع مبادئ الذكاء الاصطناعي

Nurikabe Puzzle

إعداد الطلاب :

ثراء خالد غزالي

مايا باسل عبد لله

سنا نزيه عيد

خضر عمار يوسف

ماريال حسن شاهيني

سارة أنور الجرمانى

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 برولوج.

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 ليست إحدى الخلايا المرقمة.
- كل جزيرة معزولة عن الجزر الأخرى , اي أن خلاياها لا تلتقي عمودياً أو أفقياً , 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 لتشكيل كتل من الحجم 2×2 , بينما يمكن لخلايا الجزر أن تحتوي هكذا كتل.

أولاً: تحقيق المسألة بهدف التحقق من صحة الحل :

تمثيل المسألة :

1 - نمثل الخلايا الثابتة ضمن اللوح عن طريق اسناديات، بحيث تعبر عن علاقة بين ثلاث معاملات : السطر 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 :

	3				1	
2			1			
	1			2		
		2				
1				1		6

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 الاسنادية من عدم وجود كتل من الخلايا الزرقاء بحجم 2x2 أو أكبر , و هي

معرفة وفق :

```
1 % ? - - - - -
2 validate_no_2_by_2_sea(Rows, Cols) :-
3     \+ (between(1, Rows, Row),
4         between(1, Cols, Col),
5         Row1 is Row + 1, Col1 is Col + 1,
6         solve_cell(Row, Col, blue),
7         solve_cell(Row1, Col, blue),
8         solve_cell(Row, Col1, blue),
9         solve_cell(Row1, Col1, blue)).
10 % ? - - - - -
11 no_2x2_sea:-
12     grid(R,C),validate_no_2_by_2_sea(R,C).
13 % ? - - - - -
```

حيث تقوم اسنادية ال `validate_no_2_by_2_sea(Rows,Cols)` بالمرور على جميع خلايا ال Grid و التحقق

من مجاوراتها فيما كانت تشكل كتل من الخلايا الزرقاء بحجم 2x2 :

(Xcurrent , Ycurrent)	(Xcurrent + 1 , Ycurrent)
(Xcurrent , Ycurrent + 1)	(Xcurrent + 1 , Ycurrent + 1)

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

و هي معرفة وفق :

```
1 % ? - - - - -
2 one_sea:-
3     findall((X,Y),solve_cell(X,Y,blue),BlueCells),
4     BlueCells \= [],
5     BlueCells = [(X,Y)|_],
6     sea((X,Y), Sea),
7     length(BlueCells, BlueCount),
8     length(Sea, SeaCount),
9     BlueCount == SeaCount.
10 % ? - - - - -
11 sea((X,Y),Sea):-
12     sea_helper([(X,Y)],[],Sea).
13 % ? - - - - -
14 sea_helper([], Sea , Sea).
15 sea_helper([(X,Y)|Rest] , Vis , Sea):-
16     solve_cell(X,Y,blue),
17     \+member((X,Y),Vis),
18     findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
19     append(Rest,Neighbors,NewRest),
20     sea_helper(NewRest,[(X,Y)|Vis] , Sea).
21 sea_helper([(X,Y)|Rest] , Vis , Sea):-
22     (member((X,Y),Vis);\+solve_cell(X,Y,blue)),
23     sea_helper(Rest,Vis,Sea).
24
25
```

حيث تقوم الاسنادية sea((X,Y),Sea) بعمل DFS(Deep First Search) , حيث ستحضر جميع الخلايا الزرقاء المتصلة أفقياً أو عمودياً بالخلية الزرقاء (X,Y) وتعيدها على شكل List و هذه ال List هي ال Sea , بالنسبة للاسنادية sea_helper([(X,Y)|Rest],Vis,Sea) فهي الاسنادية المساعدة للاسنادية sea وهي التي تقوم فعلياً بال DFS . بالنسبة للاسنادية التي تمثل القاعدة وهي one_sea فالمنطق الذي تتبعه هو انها تأخذ خلية زرقاء

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 :

```
1 % ? - - - - -
2 one_fixed_cell_in_island:-
3     findall((X,Y),solve_cell(X,Y,green),LandCells),
4     islands(LandCells,Islands),
5     check_islands_has_each_exactly_one_fixed_cell(Islands).
6 % ? - - - - -
7 island((X,Y),Island):-island_helper([(X,Y)],[],Island).
8 % ? - - - - -
9 island_helper([],Island,Island).
10 island_helper([(X,Y)|Rest],Vis,Island):-
11     solve_cell(X,Y,green),
12     \+member((X,Y),Vis),
13     findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
14     append(Rest,Neighbors,NewRest),
15     island_helper(NewRest,[(X,Y)|Vis] , Island).
16 island_helper([(X,Y)|Rest],Vis,Island):-
17     (member((X,Y),Vis);\+solve_cell(X,Y,green)),
18     island_helper(Rest,Vis , Island).
19 % ? - - - - -
```



```

1 % ? - - - - -
2 islands(LandCells,Islands):-islands_helper(LandCells,[],[],Islands).
3 % ? - - - - -
4 islands_helper([(X,Y)|Rest], Vis , Acc ,Islands):-
5     \+member((X,Y),Vis),
6     island((X,Y),Island),
7     append(Acc, [Island] ,NewIslands),
8     append(Vis, Island ,NewVis),
9     islands_helper(Rest,NewVis,NewIslands,Islands).
10 islands_helper([(X,Y)|Rest],Vis,Acc,Islands):-
11     member((X,Y),Vis),
12     islands_helper(Rest,Vis,Acc,Islands).
13 islands_helper([],_,Islands,Islands).
14 % ? - - - - -
15 get_island_fixed_cells_helper([],FixedCells,FixedCells).
16 get_island_fixed_cells_helper([(X,Y)|Rest],Acc,FixedCells):-
17     fxd_cell(X,Y,Z) , get_island_fixed_cells_helper(Rest,[(X,Y,Z)|Acc],FixedCells).
18 get_island_fixed_cells_helper([(X,Y)|Rest],Acc,FixedCells):-
19     \+fxd_cell(X,Y,_), get_island_fixed_cells_helper(Rest,Acc,FixedCells).
20 % ? - - - - -
21 get_island_fixed_cells(Island,IslandFixedCells):-
22     get_island_fixed_cells_helper(Island,[],IslandFixedCells).
23 % ? - - - - -
24 check_islands_has_each_exactly_one_fixed_cell([]).
25 check_islands_has_each_exactly_one_fixed_cell([Island|Islands]):-
26     get_island_fixed_cells(Island,FixedCells),
27     length(FixedCells, L),
28     L == 1,
29     check_islands_has_each_exactly_one_fixed_cell(Islands).
30 % ? - - - - -

```

حيث أن الاسنادية $island((X,Y),Island)$ تقوم بعمل DFS (Deep First Search) لتحضر جميع الخلايا الخضراء (خلايا الجزر) المتصلة أفقياً أو عمودياً بالخلية (X,Y) و تعيدها على شكل List وهذه ال List هي ال Island , اما بالنسبة للاسنادية $islands_helper([(X,Y)|Rest],Vis,Island)$ فهي اسنادية مساعدة للاسنادية $islands(LandCells,Islands)$ و هي التي تقوم فعلاً بال DFS , بالنسبة للاسنادية $islands(LandCells,Islands)$ فهي تقوم بإيجاد جميع الجزر بالاعتماد على الاسنادية $island((X,Y),Island)$ و تعيد النتيجة على شكل List Of Lists و هذه ال List هي ال Islands , و الاسنادية $islands_helper(LandCells,Vis,Acc,Islands)$ هي اسنادية مساعدة

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
 الاسنادات 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.

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

```

1 % ? - - - - -
2 island_number_equals_size:-
3     findall((X,Y),solve_cell(X,Y,green),LandCells),
4     islands(LandCells,Islands),
5     check_islands_each_has_size_equals_fixed_cell_number(Islands).
6 % ? - - - - -
7 check_islands_each_has_size_equals_fixed_cell_number([]).
8 check_islands_each_has_size_equals_fixed_cell_number([Island|Islands]):-
9     get_island_fixed_cells(Island,FixedCells),
10    FixedCells = [(_,_,L)|_],
11    length(Island, S),
12    L =:= S,
13    check_islands_each_has_size_equals_fixed_cell_number(Islands).
```

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])` التي تتحقق من المطلوب.

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

	3			1	
2			1		
	1		2		
		2			
1				1	6



```
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

	3				1	
2			1			
	1			2		
		2				
1				1		6

Violate one_fixed_cell_in_island

	3		3		1	
2			1			
	1			2		
		2				
1				1		6

Violate no_2x2_sea

	3				1	
2			1			
	1			2		
		2				
1				1		6

Violate island_number_equal_size

	3				1	
2			1			
	1			2		
		2				
1				1		6

```
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 :

```
1 :-dynamic solve_cell/3.
2 :-dynamic fxd_cell/3.
3 % Predicate to initialize the grid
4 initialize_grid(Rows, Cols, FixedCells) :-
5     retractall(solve_cell(_, _, _)),
6     retractall(fxd_cell(_, _, _)),
7     create_grid(Rows, Cols),
8     initialize_fixed_cells(FixedCells).
9 % Helper predicate to create the grid
10 create_grid(Rows, Cols) :-
11     between(1, Rows, Row),
12     between(1, Cols, Col),
13     assertz(solve_cell(Row, Col, empty)),
14     fail.
15 create_grid(_, _).
16 % Predicate to set fixed cells directly from a list
17 initialize_fixed_cells([]).
18 initialize_fixed_cells([(Row, Col, Value)|Rest]) :-
19     assertz(fxd_cell(Row, Col, Value)),
20     retract(solve_cell(Row, Col, empty)),
21     assertz(solve_cell(Row, Col, green)),
22     initialize_fixed_cells(Rest).
```

نستدعي الاسنادية 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 الاسنادية create_grid(Rows,Cols) و التي تقوم على اساس جعل جميع الخلايا باستثناء الخلايا الثابتة تحمل القيمة الابتدائية empty و نكتب solve_cell(X,Y,empty) , اما بالنسبة 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 الاسنادية 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).

```
1 % ? - - - - -
2 print_grid:-
3   grid(R,C),\+print_grid_helper(R,C).
4 % ? - - - - -
5 print_grid_helper(Rows, Cols) :-
6   between(1, Rows, Row),
7   between(1, Cols, Col),
8   (fxd_cell(Row, Col, Value) -> write(Value) ; (solve_cell(Row, Col, State) -> print_cell(State) ; write(' '))),
9   (Col == Cols -> nl; true),
10  fail.
11 print_grid(_, _).
12 % ? - - - - -
13 print_cell(green) :- write('G').
14 print_cell(blue) :- write('B').
15 print_cell(empty) :- write('.').
16 % ? - - - - -
```

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

```
1 solve_logically:-
2     solve_for_fixed_cells_with_clue_equals_one,
3     solve_for_cells_with_all_neighbors_blue,
4     seas_one_way_out,
5     solve_for_fixed_cells_that_are_one_cell_separated,
6     isolate_completed_islands,
7     solve_for_diag_adj_fxd_cells,
8     solve_for_no_2x2_blue_blocks,
9     solve_for_unreachable_cells.
10 solved:-
11     no_2x2_sea,
12     one_sea,
13     island_number_equals_size,
14     one_fixed_cell_in_island.
15 solve_logic(N):-
16     findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
17     length(EmptyCells,L),
18     L \= N,
19     solve_logically,
20     solve_logic(L).
21 solve_logic(N):-
22     findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
23     length(EmptyCells,L),
24     L == N,
25     solve_logically.
26 solve_nurikabe:-
27     solve_logic(0),
28     solve_backtracking.
29
```



```

1 solve_backtracking :-
2     % Get all empty cells
3     findall((Row, Col), solve_cell(Row, Col, empty), EmptyCells),
4     % Try to solve the puzzle
5     solve_cells(EmptyCells).
6 % ? - - - - -
7 solve_cells([]) :-
8     solved , nl , print_grid , nl , writeln("Nurikabe Solved :").
9 % Recursive case: try to solve for one cell and continue
10 solve_cells([(Row, Col)|Rest]) :-
11     % Try to color the cell blue
12     color_cell_blue(Row, Col),
13     solve_cells(Rest);
14     % If it fails, backtrack and try to color the cell green
15     color_cell_green(Row, Col),
16     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 اسنديات 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 بناءً على الاسنادية solve_logically , و تقوم باستدعاء هذه الاسنادية عدة مرات إلى أن تصل الى مرحلة لا يمكن أن تحل بهذه الاسنادية solve_logically (الاستراتيجيات 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.

```

1 % ? - - - - -
2 color_cell_blue(X,Y):-
3     retract(solve_cell(X,Y,_)),
4     assertz(solve_cell(X,Y,blue)).
5 % ? - - - - -
6 color_cell_green(X,Y):-
7     retract(solve_cell(X,Y,_)),
8     assertz(solve_cell(X,Y,green)).

```

الاسناديات المستعملة في تلوين الخلايا.

- الحل المنطقي و الاستراتيجيات المستعملة في الحل المنطقي :

كما رأينا سابقاً فإن الاسنادية solve_logically مقسمة الى مجموعة من الاسناديات التي تمثل كل منها استراتيجية حل منطقية , وهي :

1 - تلوين الخلايا المجاورة أفقياً و عمودياً للخلايا الثابتة المرفقة بالدليل 1 باللون الأزرق :

وهذا طبعاً لأن هذه الخلايا تمثل جزر منتهية و لأن الجزر لا يمكن أن تلتقي عمودياً أو أفقياً :

```

1 % ? - - - - -
2 within_the_grid(X,Y):-
3     grid(R,C) , X > 0 , X =< C , Y > 0 , Y =< R.
4 % ? - - - - -
5 neighbor((X,Y),(Xl,Y)):- Xl is X-1 , within_the_grid(Xl,Y).
6 neighbor((X,Y),(Xr,Y)):- Xr is X+1 , within_the_grid(Xr,Y).
7 neighbor((X,Y),(X,Yu)):- Yu is Y-1 , within_the_grid(X,Yu).
8 neighbor((X,Y),(X,Yd)):- Yd is Y+1 , within_the_grid(X,Yd).
9 % ? - - - - -

```

اسناديات مساعدة , الاولى within_the_grid(X,Y) للتحقق فيما إذا كانت خلية معينة (X,Y) ضمن حدود ال Grid , و الثانية neighbor((X,Y),(Xn,Yn)) لإيجاد المجاورات الأفقية و الشاقولية للخلية (X,Y) .

```

1 % ? I - Solving For Fixed Cells With Clue Equals One :
2 solve_for_fixed_cells_with_clue_equals_one:-
3     findall((X,Y),fxd_cell(X,Y,1),FixedCellsWithClueEqualsOne),
4     color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue(FixedCellsWithClueEqualsOne).
5 % ? -----
6 color_list_blue([]).
7 color_list_blue([(X,Y)|Rest]):-
8     color_cell_blue(X,Y) , print_grid , nl , nl , color_list_blue(Rest).
9 % ? -----
10 color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue([]).
11 color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue([(X,Y)|Rest]):-
12     findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
13     color_list_blue(Neighbors),color_the_neighbors_of_the_fixed_cells_with_clue_equals_one_blue(Rest).
14 % ? -----

```

تقوم الاسنادية `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 الشاقولية باللون الأزرق , الاسنادية `color_list_blue([(X,Y)|Rest])` تقوم بتلوين List الخلايا الممررة لها باللون الأزرق , واسنادية الاستراتيجية `solve_for_fixed_cells_with_clue_equals_one` تقوم بتجميع الخلايا الثابتة ذات الدليل واحد وتمديرها 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) :

```

1 % ? II - Solving For Cells With All Blue Neighbors:
2 solve_for_cells_with_all_neighbors_blue:-
3     findall((X,Y),solve_cell(X,Y,empty),CellsWithAllNeighborsBlue),
4     color_the_cells_that_all_their_neighbors_are_blue(CellsWithAllNeighborsBlue).
5 % ? -----
6 check_for_neighbors_being_blue([]).
7 check_for_neighbors_being_blue([(X,Y)|Neighbors]):-
8     solve_cell(X,Y,blue) , check_for_neighbors_being_blue(Neighbors).
9 % ? -----
10 color_the_cells_that_all_their_neighbors_are_blue([]).
11 color_the_cells_that_all_their_neighbors_are_blue([(X,Y)|Rest]):-
12     findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
13     \+check_for_neighbors_being_blue(Neighbors) , color_the_cells_that_all_their_neighbors_are_blue(Rest).
14 color_the_cells_that_all_their_neighbors_are_blue([(X,Y)|Rest]):-
15     findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
16     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).
17 % ? -----

```

تقوم الاسنادية (`check_for_neighbors_being_blue([(X,Y)|Neighbors])`) بالتحقق من كون الخلايا المجاورة (الأفقية و الشاقولية) زرقاء اللون , و بالنسبة للاسنادية (`color_the_cells_that_all_their_neighbors_are_blue([(X,Y)|Rest])`) فهي تقوم بالمرور على جميع الخلايا الفارغة (`solve_cell(X,Y,empty)` , و التحقق من كون جيرانها (الأفقية و الشاقولية) زرقاء اللون , و ذلك اعتماداً على الاسنادية (`check_for_neighbors_being_blue([(X,Y)|Neighbors])` , ثم تقوم بتلوين هذه الخلية باللون الأزرق , و ذلك في حال كانت نتيجة الاسنادية (`check_for_neighbors_being_blue([(X,Y)|Neighbors])` هي `true` , و في حال كانت نتيجتها `false` فهي تترك الخلية فارغة كما هي , وأخيراً الاسنادية التي تمثل الاستراتيجية فهي تمرر `List` الخلايا ال `empty` للاسنادية `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) :

```
1 % ? - - - - -
2 % ? III - Solving For Sea Cells With One Way Out :
3 seas_one_way_out:-
4     findall((X,Y),solve_cell(X,Y,blue),SeaCells),
5     seas_one_way_out_helper(SeaCells).
6 % ? - - - - -
7 seas_one_way_out_helper([]).
8 seas_one_way_out_helper([(X,Y)|Rest]):-
9     sea_one_way_out(X,Y) ,seas_one_way_out_helper(Rest).
10 % ? - - - - -
11 sea_one_way_out(X,Y):-
12     grid(R,C) , Xr is X + 1 , Xl is X-1 , Yu is Y-1 , Yd is Y+1 ,
13     (
14         (
15             Xl > 0,
16             solve_cell(Xl,Y,empty),
17             (solve_cell(Xr,Y,green);Xr > C),
18             (solve_cell(X,Yu,green);Yu <= 0),
19             (solve_cell(X,Yd,green);Yd > R),
20             color_cell_blue(Xl,Y)
21         )
22         ;
23         (
24             Xr <= C ,
25             (solve_cell(Xl,Y,green);Xl <= 0),
26             solve_cell(Xr,Y,empty),
27             (solve_cell(X,Yu,green);Yu <= 0),
28             (solve_cell(X,Yd,green);Yd > R),
29             color_cell_blue(Xr,Y)
30         )
31         ;
32         (
33             Yu > 0 ,
34             (solve_cell(Xl,Y,green);Xl <= 0),
35             (solve_cell(Xr,Y,green);Xr > C),
36             solve_cell(X,Yu,empty),
37             (solve_cell(X,Yd,green);Yd > R),
38             color_cell_blue(X,Yu)
39         )
40         ;
41         (
42             Yd <= R ,
43             (solve_cell(Xl,Y,green);Xl <= 0),
44             (solve_cell(Xr,Y,green);Xr > C),
45             (solve_cell(X,Yu,green);Yu <= 0),
46             solve_cell(X,Yd,empty),
47             color_cell_blue(X,Yd)
48         )
49     ),print_grid , nl , nl.
50 sea_one_way_out(_,_).
51 % ? - - - - -
```

تتحقق الاسنادية sea_one_way_out(X,Y) من كون الخلية (X,Y) خلية زرقاء , و لها مجاور واحد فارغ و بقية المجاورات ليس أي منها خلية زرقاء , ثم تقوم بتلوين ذلك المجاور الفارغ باللون الأزرق , الاسنادية seas_one_way_out([(X,Y)|Rest]) , تقوم بالمرور على جميع الخلايا الزرقاء , وتطبق الاسنادية sea_one_way_out(X,Y) عليها , و أخيراً الاسنادية التي تمثل الاستراتيجية تمرر List الخلايا الزرقاء للاسنادية 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 :

```

1 % ? -----
2 % ? IV - Solving For Complete Islands :
3 isolate_completed_islands:-
4     findall((X,Y,N),fxd_cell(X,Y,N),FixedCells),
5     isolate_completed_islands_helper(FixedCells).
6 isolate_completed_islands_helper([]).
7 isolate_completed_islands_helper([(X,Y,N)|FixedCells]):-
8     island((X,Y),Island),
9     length(Island,L),
10    L =:= N,
11    surround_the_completed_island_with_blue(Island),
12    isolate_completed_islands_helper(FixedCells).
13 isolate_completed_islands_helper([(X,Y,N)|FixedCells]):-
14    island((X,Y),Island),
15    length(Island,L),
16    L =\= N,
17    isolate_completed_islands_helper(FixedCells).
18 % ? -----
19 surround_the_completed_island_with_blue([]).
20 surround_the_completed_island_with_blue([(X,Y)|Island]):-
21     findall((Xn,Yn),neighbor((X,Y),(Xn,Yn)),Neighbors),
22     surround_the_completed_island_cell_with_blue_helper(Neighbors),
23     surround_the_completed_island_with_blue(Island).
24 % ? -----
25 surround_the_completed_island_cell_with_blue_helper([]).
26 surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]):-
27     solve_cell(X,Y,empty) , color_cell_blue(X,Y) , print_grid , nl , nl , surround_the_completed_island_cell_with_blue_helper(Neighbors).
28 surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]):-
29     \+solve_cell(X,Y,empty) , surround_the_completed_island_cell_with_blue_helper(Neighbors).
30 % ? -----

```

حيث تقوم الاسنادية surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]) بالمرور على الخلايا المجاورة الأفقية و الشاقولية لكل خلية من خلايا الجزيرة و تلوينها باللون الأزرق في حال كانت فارغة , اما بالنسبة للاسنادية surround_the_completed_island_cell_with_blue([(X,Y)|Island]) فهي تقوم بالمرور على كل خلية من خلايا الجزيرة , ثم إيجاد مجاوراتها و تمريرها للاسنادية surround_the_completed_island_cell_with_blue_helper([(X,Y)|Neighbors]) , و بالنسبة للاسنادية isolate_completed_islands_helper([(X,Y,N)|FixedCells]) فهي تمر على كل الخلايا الثابتة , 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` ليتم تحويطها بالماء , وفي حال لم يتساويا تترك الجزيرة على الحالها و تنتقل للجزيرة التالية , و أخيراً الاسنادية التي تمثل الاستراتيجية 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

```

1 % ? - - - - -
2 % ? V - Solving For Diagonally Adjacent Fixed Cells :
3 solve_for_diag_adj_fxd_cells:-
4     findall((X,Y),fxd_cell(X,Y,_),FixedCells),
5     solve_for_diag_adj_fxd_cells_helper(FixedCells).
6 % ? - - - - -
7 solve_for_diag_adj_fxd_cells_helper([]).
8 solve_for_diag_adj_fxd_cells_helper([(X,Y)|FixedCells]):-
9     Xur is X+1,
10    Yur is Y+1,
11    fxd_cell(Xur,Yur,_),
12    color_cell_blue(Xur,Y),
13    color_cell_blue(X,Yur), print_grid , nl , nl , solve_for_diag_adj_fxd_cells_helper(FixedCells).
14 solve_for_diag_adj_fxd_cells_helper([(X,Y)|FixedCells]):-
15     Xur is X+1,
16     Yur is Y+1,
17     \+fxd_cell(Xur,Yur,_),
18     solve_for_diag_adj_fxd_cells_helper(FixedCells).
19 % ? - - - - -

```

حيث تقوم الاسنادية (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 لهايتين الخليتين باللون الأزرق كما وضحنا بالشكل 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 :

```

1 % ? - - - - -
2 % ? VI - Solving For Preventing 2x2 Blue Blocks :
3 solve_for_no_2x2_blue_blocks:-
4     findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
5     prevent_2x2_blue_blocks(EmptyCells).
6 % ? - - - - -
7 if_it_will_form_2x2_blue_block_color_it_green(X,Y):-
8     Xr is X + 1,
9     Xl is X - 1,
10    Yu is Y - 1,
11    Yd is Y + 1,
12    (
13        (
14            solve_cell(Xr,Y,blue),
15            solve_cell(X,Yd,blue),
16            solve_cell(Xr,Yd,blue)
17        );
18        (
19            solve_cell(Xl,Y,blue),
20            solve_cell(X,Yu,blue),
21            solve_cell(Xl,Yu,blue)
22        )
23    ),color_cell_green(X,Y),print_grid,nl,nl.
24 if_it_will_form_2x2_blue_block_color_it_green(_,_).
25 % ? - - - - -
26 prevent_2x2_blue_blocks([]).
27 prevent_2x2_blue_blocks([(X,Y)|Rest]):-
28     if_it_will_form_2x2_blue_block_color_it_green(X,Y),
29     prevent_2x2_blue_blocks(Rest).
30 % ? - - - - -

```


حيث تقوم الاسنادية `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 الأخضر , وفي حال النفي تترك الخلية فارغة على حالها , و بالنسبة للاسنادية `prevent_2x2_blue_blocks([(X,Y)|Rest])` فهي تمر على جميع الخلايا الفارغة و تطبق عليها الاسنادية `if_it_will_form_2x2_blue_block_color_it_green(X,Y)` , و أخيراً الاسنادية التي تمثل الاستراتيجية 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 :

```

1 % ? - - - - -
2 % ? VII - solving for fixed cells that are vertically or horizontally separated by one cell :
3 solve_for_fixed_cells_that_are_one_cell_separated:-
4     findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
5     separate_fxd_cells(EmptyCells).
6 % ? - - - - -
7 fxd_cell_separator(X,Y):-
8     Xr is X + 1,
9     Xl is X - 1,
10    Yu is Y - 1,
11    Yd is Y + 1,
12    (
13        (fxd_cell(Xr,Y,_),fxd_cell(Xl,Y,_))
14        ;
15        (fxd_cell(X,Yd,_),fxd_cell(X,Yu,_))
16    ),color_cell_blue(X,Y), print_grid , nl , nl.
17 % ? - - - - -
18 separate_fxd_cells([]).
19 separate_fxd_cells([(X,Y)|Rest]):-
20     fxd_cell_separator(X,Y) , separate_fxd_cells(Rest).
21 separate_fxd_cells([(X,Y)|Rest]):-
22     \+fxd_cell_separator(X,Y) , separate_fxd_cells(Rest).
23 % ? - - - - -

```

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

8 - تلوين الخلايا التي لا يمكن الوصول إليها من أي خلية ثابتة باللون الأزرق :

```
1 % ? - - - - -
2 % ? VIII - Solving For Unreachable Cells :
3 solve_for_unreachable_cells:-
4     findall((X,Y),solve_cell(X,Y,empty),EmptyCells),
5     color_unreachable_cells_by_blue(EmptyCells).
6 % ? - - - - -
7 color_unreachable_cells_by_blue([]).
8 color_unreachable_cells_by_blue([(X,Y)|Rest]):-
9     findall((Xf,Yf,N),fxd_cell(Xf,Yf,N),FixedCells),
10    find_reachable_cells(FixedCells,[],NestedReachableCells),
11    flatten(NestedReachableCells, ReachableCells),
12    \+member((X,Y),ReachableCells),
13    color_cell_blue(X,Y),print_grid,nl,nl,
14    color_unreachable_cells_by_blue(Rest).
15 color_unreachable_cells_by_blue([(X,Y)|Rest]):-
16    findall((Xf,Yf,N),fxd_cell(Xf,Yf,N),FixedCells),
17    find_reachable_cells(FixedCells,[],NestedReachableCells),
18    flatten(NestedReachableCells, ReachableCells),
19    member((X,Y),ReachableCells),
20    color_unreachable_cells_by_blue(Rest).
21 % ? - - - - -
22 find_reachable_cells([],ReachableCells,ReachableCells).
23 find_reachable_cells([(X,Y,N)|Rest],ReachableCells,Acc):-
24    find_paths((X,Y),N,CellPaths),
25    append(CellPaths,ReachableCells,NewReachableCells),
26    find_reachable_cells(Rest,NewReachableCells,Acc).
27 % ? - - - - -
28 find_path((X,Y),C,Vis,Path):-
29    C > 0,
30    neighbor((X,Y),(Xn,Yn)),
31    \+member((X,Y),Vis),
32    C1 is C-1,
33    find_path((Xn,Yn),C1,[(X,Y)|Vis],Path).
34 find_path((_,_),0,Path,Path).
35 find_paths((X,Y),L,Paths):-
36    setof(Path,find_path((X,Y),L,[],Path),Paths).
37 % ? - - - - -
```

حيث تقوم الاسنادية $\text{find_path}((X,Y),C,\text{Vis},\text{Path})$ بإيجاد مسار من الخلايا طوله C و ينطلق من الخلية (X,Y) , اما الاسنادية $\text{find_paths}((X,Y),L,\text{Path})$ فهي تقوم و بالاعتماد على الاسنادية السابقة بإيجاد جميع المسارات الممكنة التي طولها L و تبدأ من الخلية (X,Y) , بالنسبة للاسنادية $\text{find_reachable_cells}([(X,Y,N)|\text{Rest}],\text{Reachable Cells},\text{Acc})$ فهي تقوم بإيجاد كل المسارات الممكنة لكل خلية ثابتة و التي طولها الدليل المرفق بالخلية الثابتة , و تعيد List بهذه المسارات كلها , بالنسبة للاسنادية $\text{color_unreachable_cells_by_blue}([(X,Y)|\text{Rest}])$ فهي تمرر List الخلايا الثابتة للاسنادية $\text{find_reachable_cells}([(X,Y,N)|\text{Rest}],\text{Reachable Cells},\text{Acc})$ و بعدها تقوم بعمل flatten لل List الراجعة من هذه الاسنادية والتي تمثل List من المسارات (List Of Lists) وهي ال $\text{NestedReachableCells}$, 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 تمرر هذه الاسنادية على كل الخلايا الفارغة و تتحقق فيما اذا كانت الخلية تنتمي لل ReachableCells ففي حال الإيجاب تظل 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 $\text{color_unreachable_cells_by_blue}([(X,Y)|\text{Rest}])$.

- 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
1GB1B6

Nurikabe Solved :)
true .
```



		3				1	
2			1				
	1			2			
		2					
1				1			6

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.
```

```
BBBBGGBB
2B1B2B2
GBBBBBG
BB1B1BB
GBBBBBG
2B1B1B2
BBBBBBB

Nurikabe Solved :)
true .
```



2		1		2		2
		1		1		
2		1		1		2

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

Thank

You