جامعة دمشق

كلية الهندسة المعلوماتية

السنة الثالثة – قسم الذكاء الصنعي

مشروع عملي مبادئ الذكاء الصنعي

مجموعات الطلاب:4 طلاب على الأقل إلى 6 طلاب على الأكثر في كل مجموعة. ويمكن أن يكونوا من فئات مختلفة.

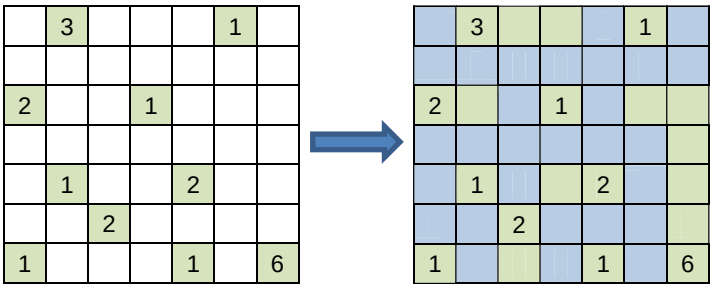
موعد التسليم والمقابلات: تسليم الوظيفة يوم الخميس 2024/6/27. وتتم المقابلات في نفس اليوم ابتداءً من الساعة التاسعة صباحاً بعد التسجيل على الاستمارة على الرابط [1]بحيث يسجل من كل مجموعة طالب واحد فقط. لا تمنح علامة للطلاب الذين لا يحضرون المقابلة.

طريقة التقديم: تقرير مطبوع يحتوي الشرح والاستنتاجات والبرامج، إضافة إلى مقابلة مخصصة للعرض والتقييم.

نوريكابي (Nurikabe)

لعبة نوريكابي من ألعاب الأحاجي التي يلعبها لاعب واحد والتي تعتمد على المنطق، سنقوم بكتابة برنامج يقوم بحل هذه اللعبة باستخدام لغة البرمجة برولوغ.

يمكنك تجريب هذه اللعبة على هذا الرابط



[1] يجب ألا يقل عدد الطلاب عن 4 ولا يزيد عن 6 ولا يقبل تسجيل مجموعات لا تحقق شرط العدد.

آخر موعد للتسجيل على الاستمارة هو الجمعة 6/14

يمكن تعديل المجموعات بعد التسجيل على الرابط حتى تاريخ الجمعة 6/14

لا يقبل أبداً تغيير مجموعات الطلاب أو إضافة طلاب على المجموعات بعد هذا التاريخ.

تتألف اللعبة من جدول مقسم إلى خلايا، يمكن أن يكون للجدول قياسات مختلفة. بعض هذه الخلايا تحوي 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، ولا تحوي الرقعة على جزر خلاياها ليست إحدى الخلايا المرقمة.
- كل جزيرة معزولة عن الجزر الأخرى، أي أن خلاياها لا تلقي عمودياً أو افقياً، 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 كتلة من 2×2 خلية أو أكثر، بينما يمكن لخلايا الجزر أن تحوي هذه الكتل.

أولاً: تحقيق المسألة بهدف التأكد من صحة حل

تمثيل المسألة

سنقترح طريقة لتمثيل المعطيات، التزم في حلك بهذه البنية **حصراً**[2]. تتيح هذه البنية إمكانية حل أي لعبة موصفة قابلة للحل (ليس فقط المثال السابق)

1- نمثل الخلايا الثابتة ضمن اللوح عن طريق اسناديات، بحيث تعبر عن علاقة بين ثلاث معاملات: السطر والعمود وقيمة الرقم فيها، بحيث يكون لدينا حقيقة من أجل كل خلية ثابتة ضمن الجدول.

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

من أجل المثال السابق يكون لدينا 10 حقائق على الشكل التالي:

```
fxd_cell(1,2,3).
fxd_cell(1,6,1).
fxd_cell(3,1,2)

...etc
```

[2] في حال استخدامك لبنية مختلفة لا تقدم المشروع 😳

2- نمثل خلايا الحل بإسناديات (علاقة بين سطر وعمود ولون) مولدة بشكل ديناميكي خلال الحل

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

من أجل المثال السابق يكون لدينا مثلاً

```
solve_cell(1,1,blue).
solve_cell(1,3,green).
solve_cell(1,4,green).
...etc
```

خلايا الحل يجب أن تكون مولدة آلياً بشكل ديناميكي خلال لعب اللعبة، ولكن مبدئياً يمكنك إضافة حقائق ثابتة تمثل خلايا الحل في هذا الجزء من الوظيفة، ثم إزالتها من الكود في القسم الثاني من الوظيفة.

والمطلوب: التحقق من كون رقعة ما تمثل حلاً صحيحاً لمسألة

لنفرض أنه لدينا رقعة محلولة، أي أنه لديك مجموعة حقائق `fxd_cell` ومجموعة حقائق `solve_cell` تمثل حلاً للمسألة، وتريد التحقق فيما إذا كان هذا الحل هو حل صحيح، أي يحقق جميع شروط المسألة.

للمساعدة سنستعرض مجموعة من الإجرائيات التي يمكن أن تستخدمها في حلك، يمكنك جمع هذه الإجرائيات في إجرائيات أكبر ولك الحرية في طريقة تحقيقها (أو عدمه في حال أحببت أن تحل بطريقة أخرى **مع الالتزام بالبنية المقترحة**).

1. إيجاد جميع جوارت خلية بشرط أن يكون له نفس اللون
   مثلاً جوار الخلية (1,2) والذي له نفس اللون هو فقط الخلية (1,3)
   وجوار الخلية (2,3) والذي له نفس اللون هو الخلايا (2,2) و (2,4) و (3,3)
2. إيجاد مجموعة الخلايا من نفس اللون المتجاورة بالسطر أو العمود والتي تشكل معاً مجموعة (جزيرة أو بحر) ابتداءً من خلية معينة ووضعها في قائمة بالاستعانة بالإجرائية السابقة.
   مثلاً إيجاد مجموعة جوارات الخلية (1,1) في الحل النهائي يعطي قائمة بجميع الخلايا الزرقاء في الرقعة
   أما إيجاد جوارات الخلية (1,4) يعطي قائمة بالخلايا (1,2) و (1,3) و (1,4)).

3. المطلوب منك في هذا الجزء من المسألة كتابة إجرائية `solved` التي تتحقق أن الخلايا تحقق قواعد اللعبة وذلك بإيجاد جميع الجزر والبحور والتأكد من كونها لا تخرق القواعد حيث يكفي أن يخل الحل بقاعدة واحدة حتى يعتبر غير مقبول. يمكن تقسيمها إلى قواعد جزئية على النحو التالي:
   1. إجرائية `one_sea` تتحقق أن البحر وحيد.
   2. إجرائية `no_2_by_2_sea` البحر لا يحوي خلايا تشكل كتلة من 2×2 خلية[3].
   3. إجرائية `one_fixed_cell_in_island` تتأكد من أن الجزر تحوي خلية ثابتة واحدة.
   4. إجرائية `island_number_equals_size` تتأكد أن عدد الخلايا في الجزيرة الواحدة يساوي الرقم في الخلية الثابتة.

بعد كتابتك لهذه القواعد، ما عليك سوى كتابة قاعدة تقوم بتجميعها للتحقق من الشروط جمعياً مثلاً على الشكل التالي:

```
solved:- one_sea,
         no_2_by_2_sea,
         one_fixed_cell_in_island,
         island_number_equals_size.
```

لاحظ أن هذه القاعدة والقواعد التي تحتويها لا تأخذ متحولات حيث أنها ليست بحاجة إليها.

ثانياً: حل المسألة باستخدام القواعد

بدايةً قم بتعليق جميع الحقائق التي تمثل الحل والتي قمت بإضافتها لتجريب الطلب السابق، حيث أن مهمة البرنامج هو إيجاد هذه الحقائق أثناء الحل بشكل ديناميكي.

يتبع لاعبو Nurikabe قواعد تمثل استراتيجيات للحل تأتي من خبرتهم بلعب هذه اللعبة، ويقومون بملء الرقعة تدريجياً بتكرار الحل بهذه القواعد حتى إيجاد الحل كاملاً (في حال كانت هذه القواعد كافية لحل المسألة). كلما كانت القواعد أكثر تعقيداً كلما كان برنامجك قادراً على حل مسائل أصعب.

الروابط التالية تشرح مجموعة استراتيجيات يمكنك تطبيقها أثناء الحل، حقق ما تستطيع منها:

- http://www.conceptispuzzles.com/index.aspx?uri=puzzle/nurikabe/tutorial
- http://www.conceptispuzzles.com/index.aspx?uri=puzzle/nurikabe/techniques

[3] يمكنك حل هذه الإجرائية بطريقتين الأولى هي المرور على خلايا البحر (بعد تجميعها في سلسلة بشكل مرتب) والتأكد من عدم وجود جوارات تشكل كتلة غير مقبولة، أو يمكنك حلها بطريقة نقض الفرض، أي أن تكتب قاعدة توجد كتلة بحر 2×2 ثم تقوم بنفي هذه القاعدة.

مساعدة

ستحتاج مجموعة قواعد مساعدة منها:

1. تهيئة اللعبة: والذي يتضمن حذف كل الإسناديات الديناميكية الموجودة مسبقاً (من تشغيل سابق للعبة)[4] .
2. طباعة الرقعة: (ننصحك بكتابه هذه القاعدة بداية لتتأكد من شكل الرقعة عند كل حل) يجب أن تتم الطباعة بعد كل خطوة لعب، بحيث تطبع الرقعة سطراً سطراً[5]،قم بطباعة الخلايا بشكل مرتب (محاطة بفراغات أو رموز...) لتسهل رؤيتها.
3. حل اللعبة: تبدأ اللعبة بالتهيئة ثم تطبيق القواعد مجموعة من القواعد المكررة أي التي تكرر[6] مجموعة الخطوات تمثل كل منها استراتيجية أثناء الحل، ثم تتحقق من انتهاء الحل، يجب أن تقوم بطابعة الرقعة بعد كل دورة منها للتأكد من تقدم الحل (في حال لم يتم إضافة أي حقيقة جديدة قم بإنهاء الحل بالفشل كي لا يعلق البرنامج أثناء تنفيذ حلقة لا نهائية).

تحدي 😊

إذا استطاع برنامجك حل المسائل من المستوى المتقدم ستنال علامة أعلى من العلامة الكاملة (يمكن أن تعوض نقصاً في التقييمات) هذا يتطلب منك استخدام استراتيجيات حل أكثر تعقيداً.

---

[4] إن استدعاء الحل أكثر من مرة بدون إعادة تشغيل برولوغ يؤدي إلى أن القواعد التي تم إضافتها عند التنفيذ السابق لم تحذف، وقد تؤدي إلى عمل البرنامج بشكل خاطئ، يمكنك كتابة قاعدة clear تحذف جميع الحقائق الديناميكية التي قمت بإضافتها باستخدام retractall واستدعائها قبل كل تنفيذ.

[5] هذه القاعدة (وقواعد اخرى في هذه المسألة) تتطلب المرور على جميع عناصر الجدول، هذا يشبه تطبيق حلقة for على الأسطر والأعمدة، طبعاً هذه التعليمة غير متوفرة في prolog ولكن يمكن كتابة القواعد على الشكل التالي لتقوم بنفس الأثر:

```
% rule that needs looping (e.g. print...)

loop_for_some_reason :- get_row(Row), get_col(Col) %predicate that generates
numbers from 1 to the size of the grid…
                        process(Row, Col, ..) %do something with one solution
                        ,fail. %so the rule will fire again!

loop_for_some_reason:- true. %so the rule will not return false when done
```

[6] يمكنك تحقيق هذا التكرار بنفس فكرة المرور على الحلقات الموجود في الطباعة.

توجيهات :

- ننصحك بتجريب القواعد كل على حدة، وعدم كتابتها كلها معاً وذلك لتسهل عملية اكتشاف الأخطاء، يمكن الاستعانة بالتعليمة trace و gtrace التي تمكنك من القيام بـ debugging للقواعد التي تكتبها.
- يجب أن تقوم بتجهيز أكثر من رقعة لتجريب الحل عليها، يمكنك الحصول على رقع مختلفة من الروابط السابقة.
- لا ننصح بتقسيم طلبي المسألة بين طلاب المجموعة، وذلك لتعلق الحلول والقواعد ببعضها، وإنما يجب على جميع أفراد المجموعة وضع قواعد المسألة معاً، ثم يمكن تقسيم تحقيقها برمجياً بينهم، ويجب توضيح تقسيم العمل في التقرير.
- يحوي التقرير على شرح لطريقة تمثيلك للمسألة والطريقة التي اتبعتها في الحل موضحاً فيه جميع القواعد والإسناديات مع شرح بسيط وواضح لكل منها، بالإضافة إلى أمثلة اختبار لتوضيح الفكرة عند الحاجة. يجب مراعاة ترتيب التقرير، وجودة الحل، وكتابة المتحولات بأسماء معبرة، استخدم قياس خط 12 وتنسيق مناسب لا يحوي على فراغات.

  في حال كان التقرير هو نسخة عن الكود بدون أي شرح كافي للقواعد والحقائق فلن تنال علامته.
- يحدد وقت المقابلة قبل التسليم بيوم واحد، ننصح بالتحضير للمقابلة كي تتم خلال 5 دقائق، يتم خلالها عرض العمل الشخصي بمشاركة جميع طلاب المجموعة والنتائج العملية التي تم التوصل إليها من خلال مثال اختبار مجهز مسبقاً للمسألة كاملة ولكل خطوة جزئية على حدة. تأكد من أن يكون التقرير جاهزاً ومطبوعاً قبل المقابلة وإلا لن تنال علامة عليه.
- على جميع طلاب المجموعة المشاركة في حل الوظيفة، ويجب أن يكونوا على علم كامل بكافة طلبات المسألة، والطلاب الذين لم يشاركوا في الحل لن ينالوا أي علامة.
- ميثاق الشرف (honor code)
  - أية وظيفة منقولة من جهة ثانية أو منقول منها أو مأخوذة من الإنترنت ستنال علامة الصفر حتماً،
  - سيتم التأكد من تشابه الأكواد باستخدام برمجيات لكشف الغش، ثم مراجعتها يدوياً، ولن يتم الرجوع إلى الطلاب في حال اكتشاف أي تشابه، سيتم حذف علامة الجزء المتشابه من المجموعتين الناقلة والمنقول منها، ولن يتم إعادة العلامة للمجموعة صاحبة الكود حتى لو لم يكن الجميع على علم بحالة الغش الحاصلة.
  - لا يمكن لمجموعتين التشارك بالحل معاً لا بالفكرة ولا بالأكواد أبداً.
  - لا يجوز الاستعانة بأي أدوات ذكاء صنعي توليدي (Generative AI) لتوليد الكود (مثل Chatgpt)، لإن الغرض من المشروع هو تقييم فهم الطلاب، وليس قدرتهم على جعل الذكاء الصنعي يقوم بالعمل نيابة عنهم. بالتالي أي جزء من الكود لا يستطيع الطلاب شرحه كاملاً بالتفصيل أو يستخدم تقنيات لم يتم

شرحها ضمن المقرر أو تشابه مع مجموعة أخرى (لاستخدامها نفس الطريقة في التوليد) سيعتبر غشاً وستحذف علامته لهذا السبب.

- بعد الانتهاء من المقابلة تسلم نسخة من الكود بلاحقة pl. بدون ضغط واسم الملف بأسماء الطلاب باللغة العربية للمهندس الذي قابلت عنده المجموعة (على فلاشة المهندس). ثم قم برفع التقرير والكود نفسه على الرابط (التزم تمامًا بطريقة التسليم المذكورة وإلا يعتبر تسليمك لاغياً، لا تقم بضغط الملفات قبل رفعها).

مدرسو العملي