BloxorGT

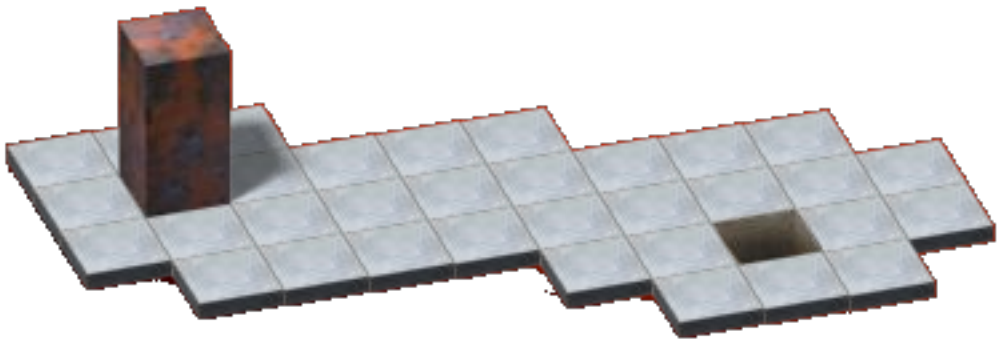
Paradigmes i Llenguatges de Programació.

March 9, 2020

Introducció

El joc Bloxorz és un típic joc trencaclosques que es pot enmarcar en els clàssics problemes de la intel·ligència artificial de *planificació*: donat un mon amb unes unes accions que en poden canviar l'estat, decidir si existeix una seqüència d'accions que ens porten de l'estat inicial a l'estat final.

A Bloxorz el mon ens ve donat com a una plataforma de caselles suspesa en un espai buit pel qual s'ha de desplaçar un bloc format per dos cubs unitaris enganxats, de manera que si el bloc està dret ocupa una casella i si està estirat dues, fins a fer-lo caure dins una casella foradada final o meta. El nostre joc, el `BloxorGT`, és una generalització del Bloxorz on el bloc a moure serà un prisma rectangular que pot estar format per un o més cubs unitaris enganxats, de manera que tindrà una base (dimensió x al tauler), una altura (dimensió y al tauler) i un gruix (tercera dimensió z indicant quant s'alça el prisma sobre el tauler). Fixeu-vos que el bloc del Bloxorz és un cas particular del nostre `BloxorGT` on el bloc, tal com apareix a la figura, seria un bloc $1\times1\times2$. El bloc el podem desplaçar "empenyent-lo" amunt, avall, esquerra o dreta, així el farem rodolar pel mon de cara en cara del prisma. Pel tal que el bloc no caigui a l'abisme ha d'arrepenjar **tota** la superfície al tauler. L'objectiu serà obtenir la seqüència de passos que porta el bloc de la posició inicial al forat destinació, sense caure a l'abisme. En el joc original aquest forat final porta al següent nivell i correspon a una sola casella. En el nostre cas, el forat final serà un conjunt de caselles que formen un rectangle de les dimensions que volguem i perque el bloc hi pugui entrar s'hi haurà d'*arrepenjar* totalment. Fixeu-vos que, suposant que tenim un bloc d'$1\times1\times2$ i tenint un forat final que ocupa dues caselles, tant hi podrà entrar dret (arrepenjant-se totalment en només una casella final) com tombat (arrepenjant-se totalment en les dues caselles finals). Assumim també, que el bloc es pot arrepenjar parcialment en caselles de forat final i en caselles normals sense caure al buit ni acabar el nivell.



Per a familiaritzar-vos amb el joc Bloxorz (és a dir, la versió on el bloc només pot ser d'$1\times1\times2$) podeu visitar: `http://www.coolmath-games.com/0-bloxorz`

1 Objectius

L'objectiu d'aquesta pràctica és que sigueu capaços de resoldre problemes `BloxorGT` donant-ne la seqüència de moviments més curta, si existeix, o dient que no te solució altrament. Es voldrà que oferiu una representació de l'evolució del tauler fins a l'estat final, d'acord amb els moviments de la solució trobada.

1.1 L'entrada

El mon, que serà sempre rectangular, estarà descrit mitjançant fitxers de text com el següent que primer ens indica el gruix del bloc, després el nombre de files, columnes i la configuració del tauler:

```
2
6
10
1110000000
1S11110000
1111111110
0111111111
0000011G11
0000001110
```

`0` representa el buit, `1` les caselles "normals", `S` les caselles de sortida on s'arrepenja el bloc i `G` les caselles finals o meta.

S'assumeix que les caselles de sortida també són normals, és a dir, que no n'hi ha cap que sigui buida.

1.2 Sortida

Per aquesta entrada, la sortida hauria de ser quelcom com:

```
Inici          1 dreta        2 dreta        3 avall
1110000000     1110000000     1110000000     1110000000
1B11110000     11BB110000     1111B10000     1111110000
1111111110     1111111110     1111111110     1111B11110
0111111111     0111111111     0111111111     0111B11111
0000011G11     0000011G11     0000011G11     0000011G11
0000001110     0000001110     0000001110     0000001110


4 dreta        5 dreta        6 dreta        7 avall
1110000000     1110000000     1110000000     1110000000
1111110000     1111110000     1111110000     1111110000
11111B1110     111111B110     1111111B10     1111111110
01111B1111     011111B111     0111111B11     0111111111
0000011G11     0000011G11     0000011G11     0000011B11
0000001110     0000001110     0000001110     0000001110

                                             Si, en 7 passos!!!
```

Per indicar moviments amunt o esquerra farem servir les mateixes paraules `amunt` i `esquerra`. En cas de no trobar solució direm `No hi ha solució!!!`.

Tipus i funcions obligatoris

La vostra pràctica **haurà d'implementar alguns tipus i funcions que indiquem tot seguit, obligatòriament**. Aquests us han de servir de guia per a completar-la.

Tipus

- `Tauler`, tindrà la informació necessària sobre les caselles que configuren el mon.
- `Bloc`, tindrà la informació de l'estat del bloc, per exemple, la casella 0,0 (és a dir la de més esquerra i més envall que ocupa) i les seves dimensions x, y i z, (o altres representacions que se us acudeixi).
- `Moviment`, que pot ser `Amunt`, `Avall`, `Dreta` i `Esquerra`.

Penseu si heu de posar la informació de les caselles d'origen i les de final en algun d'aquests tipus o si no cal. També penseu si us convé una noció d'igualtat entre `Blocs` o entre `Tauler`.

Seria interessant tenir un tipus `Estat` *on hi hagués el* `Bloc` *i el* `Tauler`*? seria eficient? Us pot caler un tipus* `Posicio`*? ...*

Funcions

- `posBloc`, ens ha de donar les posicions que ocupa el bloc.
- `casellaBuida`, ens ha de dir si la posició que preguntem te casella o està buida.
- `resolt`, ens ha de dir si el bloc ja ha arribat a la meta o no.
- `sortida`, ens ha de crear un tauler i bloc a l'estat inicial.
- `esLegal`, ens ha de dir si tal com està el bloc es possible o no fer cert moviment en cert tauler, és a dir, si aquest moviment és legal o fa caure el bloc al buit.
- `legals`, ens ha de tornar els moviments que siguin legals en l'"estat" actual.
- `mou`, ha de fer efectiu un moviment (si és legal) i portar-nos a l'"estat" que en resulti.

Fixeu-vos que no us he donat la signatura de les funcions. Depen molt de les decisions que feu amb els tipus...

Resoledor i Joc

Caldrà que feu un programa que demani a l'usuari el nom del fitxer on hi ha el mon i el mostri. Llavors l'usuari tindrà l'opció de jugar o de resoldre. Per jugar haurà de llegir les comandes (o llegir tecles) i anirà mostrant com va canviant el mon, acabant la partida si el bloc cau o arriba al final.

Per a resoldre caldrà fer una cerca en amplada. Una jugada vàlida seria una seqüència d'estats que comença amb l'estat inicial juntament amb la llista de moviments legals que han provocat canvis d'estat. Una solució seria una jugada vàlida que acaba amb el bloc totalment sobre les caselles finals (esta final).

La proposta que us faig per a resoldre el problema és la de la "bassa d'oli": a partir de l'estat inicial, anar generant totes les jugades legals que es poden fer de l'estat actual amb un sol moviment, si hem arribat al final ja està, si no, seguim "escampant la taca d'oli" a partir dels nous moviments legals. Compte, caldrà evitar revisitar estats on ja haguéssim arribat amb una jugada més curta ja que això ens portaria a cicles, comprometent la nostra cerca de la solució. Fixeu-vos que aquesta "taca d'oli" està suggerint una cerca en amplada enlloc de en profunditat.

Si algú, a més a més d'aquesta aproximació per a solucionar el problema, vol fer-ne alguna altra, serà ben valorada.

Es demana

Com ja hem dit d'objectiu serà que feu un resoledor de `BloxorGT` amb Haskell i també un joc interactiu (que el podeu fer amb mode text). Per al joc interactiu podeu imaginar que el fitxer de text conté els diferents nivells seguits un darrera l'altre.

Fer això bé permet obtenir 9 punts dels 10 que val la pràctica. Per aconseguir el punt restant, caldrà que feu alguna extensió, que de fet ja apareixen a `Bloxorz`:

- (0,5 punts) *Caselles fràgils*: hi ha un tipus de casella on quan el bloc s'hi arrepengi no pot tenir un gruix major que 1 (en el cas del bloc d'1×1×2, això vol dir que no es pot posar dret en una casella com aquesta). Suposeu que aquesta modificació es faria en l'entrada marcant aquestes caselles amb dosos enlloc de amb uns.

- (1 punt) *Ponts*: hi ha caselles especials (marcades amb una X per exemple), que funcionen com a interruptors i pugen o baixen "ponts", és a dir, fan aparèixer caselles on abans hi havia el buit. Típicament es fan servir per conectar illes de caselles. Es podrien especificar al mapa marcant amb una majúscula la casella que fa d'interruptor (per exemple `X`) i amb minúscules les caselles que fan de "pont fantasma" (per exemple `x`. Fixeu-vos que com que la `S`, la `G` i la `B` estan reservades, podríem tenir com a molt 24 (27-3) interruptors i ponts diferents.

- Si algú te alguna extensió alternativa també la pot proposar. Per exemple fer un entorn de joc que no sigui només text.

Es valorarà positivament la definició i us d'uns bons tipus, l'ús adequat de funcions d'ordre superior i la programació "elegant". També es valorarà l'eficiència ja que l'espai de cerca pot arribar a ser molt gros i problemes massa grans poden ser difícils de resoldre...

Lliurament

- Les pràctiques **S'HAN** de fer en equips de dos.

- Cal que documenteu el codi.

- S'haurà de lliurar un document **IMPRÈS** amb el codi comentat i les execucions que es demanen mostrant els resultats obtinguts (captures de pantalla). En concret **el document caldrà que tingui els següents apartats**:

  1. Descripció del problema resolt i abast de la vostra solució: breument definir el problema que heu resolt, incloent les possibles extensions o limitacions de la vostra solució.
  2. Tipus utilitzats: codi i descripció dels tipus que heu utilitzat, així com de les instanciacions de classes que n'hagueu fet.
  3. Codi comentat: codi breument comentat, ben indentat i llegible.
  4. Particularitats del codi: descripció detallada de les funcions més rellevants.
  5. Jocs de prova: descripció dels jocs de prova i "pantallassos" d'us.
  6. Referències de bibliografia i/o llibreries usades si s'escau.

- **Les entregues molt probablement seran presencials** per poder avaluar la implicació en la pràctica per part dels membres de l'equip. També es possible que per acabar d'avaluar se us faci fer alguna modificació de la pràctica *in situ*.

- Data de lliurament (per correu electrònic i en paper): **20 d'abril.**