« به نام او »

آموزش شبیه سازی دو بعدی فوتبال

نویسندگان : محمدعلی میرزایی - علی یعقوبی

مرجع روبوکاپ ایران

www.iranrcss.com

قسمت هفتم

مباحث این جلسه :

- ادامه آموزش بیس UVA
- آموزش هوش مصنوعی(Artificial Intelligent) ، حل مساله با روش جستجو

در جلسه قبل بخش اول کد زیر را با هم بررسی کردیم . در این جلسه به ادامه آن می پردازیم :

```
else if( WM->getFastestInSetTo( OBJECT_SET_TEAMMATES, OBJECT_BALL, &iTmp )
             == WM->getAgentObjectType()  && !WM->isDeadBallThem() )
  {                                                  // if fastest to ball
    Log.log( ۱۰۰, "I am fastest to ball; can get there in %d cycles", iTmp );
    soc = intercept( false );                        // intercept the ball

    if( soc.commandType == CMD_DASH &&               // if stamina low
        WM->getAgentStamina().getStamina() <
           SS->getRecoverDecThr()*SS->getStaminaMax()+۲۰۰ )
     {
       soc.dPower = ۳۰.۰ * WM->getAgentStamina().getRecovery(); // dash slow
       ACT->putCommandInQueue( soc );
       ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
     }
     else                                             // if stamina high
     {
       ACT->putCommandInQueue( soc );                 // dash as intended
       ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
     }
    }
    else if( posAgent.getDistanceTo(WM->getStrategicPosition()) >
                  ۱.۵ + fabs(posAgent.getX()-posBall.getX())/۱۰.۰)
                                                   // if not near strategic pos
    {
```

```
۲۵-       if( WM->getAgentStamina().getStamina() >     // if stamina high
۲۶-                                SS->getRecoverDecThr()*SS->getStaminaMax()+۸۰۰ )
۲۷-       {
۲۸-         soc = moveToPos(WM->getStrategicPosition(),
۲۹-                              PS->getPlayerWhenToTurnAngle());
۳۰-         ACT->putCommandInQueue( soc );            // move to strategic pos
۳۱-         ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۳۲-       }
۳۳-       else                                         // else watch ball
۳۴-       {
۳۵-         ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT_BALL ) );
۳۶-         ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۳۷-       }
۳۸-     }
۳۹-     else if( fabs( WM->getRelativeAngle( OBJECT_BALL ) ) > ۱۰ ) // watch ball
۴۰-     {
۴۱-       ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT_BALL ) );
۴۲-       ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۴۳-     }
۴۴-     else                                          // nothing to do
۴۵-       ACT->putCommandInQueue( SoccerCommand(CMD_TURNNECK,۰.۰) );
۴۶-   }
```

در خط ۴ از تابعی به نام log.log استفاده شده است . هنگامی که یک بازی اجرا می شود ، این لاگ ها در فایلی ذخیره می شود . کاربرد این لاگ ها این است که شما بعد از اتمام بازی می توانید به لاگ خوانی بپردازید و مشکلات تیمتان را به راحتی حل کنید . با این کار شما متوجه می شوید آیا تابع مورد نظرتان اجرا شده است یا خیر.

در خط بعدی تابع intercept به کار رفته است . این تابع همانطور که از اسمش پیدا است مسئول تصاحب توپ است . این تابع آرگومانی از نوع بولین دارد که اگر true باشد ، به این معناست که بازیکن اجرا کننده دستور، دروازه بان است و اگر false باشد ، سایر ایجنت ها میباشد . این تابع در بیس uva دارای مشکلاتی است که توصیه می شود ، حتماً دوباره این تابع را پایه ریزی کنید .

از خط شماره ۷ تا ۲۰ برای مدیریت stamina بازیکن می باشد . این stamina Manager بسیار ساده می باشد . حال خط به خط به توضیح آن می پردازیم :

```
۷ -     if( soc.commandType == CMD_DASH &&              // if stamina low
۸ -         WM->getAgentStamina().getStamina() <
۹ -            SS->getRecoverDecThr()*SS->getStaminaMax()+۲۰۰ )
```

در ابتدای این شرط بررسی می شود که آیا `soc.commandType == CMD_DASH` می باشد یا خیر ؟ حال این چیست؟

همانطور که قبلا اشاره شده بود soc یک متغیر از نوع SoccerCommand می باشد . در بیس UVA ، SoccerCommand در انتها به یکی از CommandT زیر تبدیل می شود . حال ما با استفاده از دستور soc.commandType آن CommandT را از بیس می گیریم . حال این CommandT ها چه هستند ؟

تعریف CommandT - درفایل SoccerTypes.h می باشد - را اینجا قرار می دهم . توضیحات خود بیس گویای همه چیز است :

```
/*!The CommandT enumeration contains the different types for the SoccerCommand
   that are possible. */
enum CommandT {
  CMD_ILLEGAL,      /*!< illegal command */
  CMD_DASH,         /*!< dash command (player only)         */
  CMD_TURN,         /*!< turn command (player only)         */
```

```
  CMD_TURNNECK,     /*!< turn_neck command (player only)    */
  CMD_CHANGEVIEW,   /*!< change view command (player only)  */
  CMD_CATCH,        /*!< catch command (goalie only)        */
  CMD_KICK,         /*!< kick command (player only)         */
  CMD_MOVE,         /*!< move command                       */
  CMD_SENSEBODY,    /*!< sense_body command (player only)   */
  CMD_SAY,          /*!< say command                        */
  CMD_CHANGEPLAYER, /*!< change_player command (coach only) */
  CMD_ATTENTIONTO,  /*!< pay attention to specific player   */
  CMD_TACKLE,       /*!< tackle in current body direction   */
  CMD_POINTTO,      /*!< point arm towards a point on field */
  CMD_MAX_COMMANDS  /*!< maximum number of commands         */
} ;
```

در ادامه شرط دیگری نیز بررسی می شود . این شرط این است :

```
WM->getAgentStamina().getStamina() < SS->getRecoverDecThr()*SS->getStaminaMax()+۲۰۰
```

تابع `WM->getAgentStamina().getStamina()`، stamina ی بازیکن را به ما می دهد . stamina عددی است از ۰ تا ۸۰۰۰ که برای ما انرژی بازیکن را مشخص می سازد . هر چه این عدد بالاتر باشد، انرژی بازیکن بالاتر است . تابع `SS->getRecoverDecThr()` به ما متغیری را به نام `dRecoverDecThr` به ما می دهد که خود بیس این متغیر را اینگونه تعریف می کند :

```
double dRecoverDecThr;    /*!< recover_dec_thr: percentage of stamina_max
                                below which player recovery decreases      */
```

و تابع `SS->getStaminaMax()` به ما متغییری را به نام `dStaminaMax` به ما می دهد که خود بیس این متغیر را اینگونه تعریف می کند :

```
double dStaminaMax;       /*!< stamina_max: maximum stamina of a player    */
```

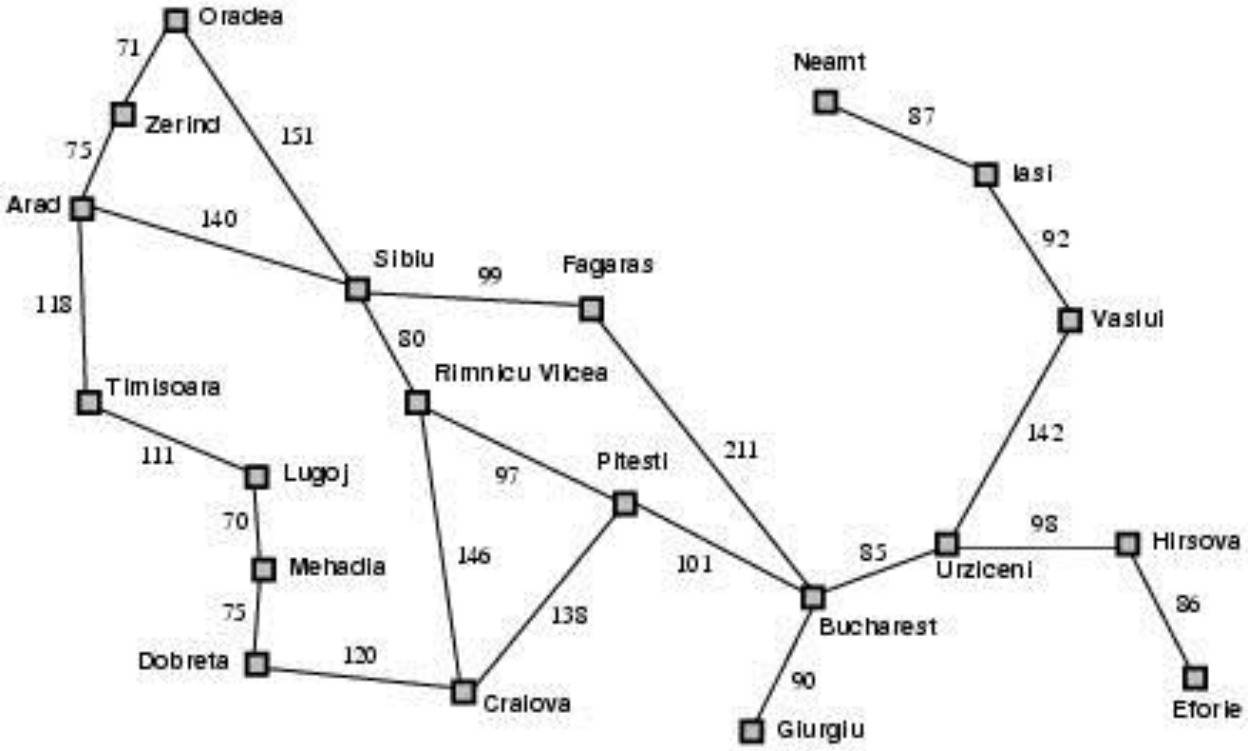
ادامه این مبحث را در جلسه ی بعد دنبال میکنیم .

هوش مصنوعی

حل مساله به روش جستجو

در جلسه پیش، فهرستی از روش های حل مسأله را مشاهده کردیم. هم اینک میخواهیم بحث را با ارائه مثالی گسترده تر کنیم.

مثال نقشه رومانی :



- **صورت مسأله: رفتن از آراد به بخارست**
- **فرموله کردن هدف: رسیدن به بخارست**
- **فرموله کردن مسئله:**
  - **وضعیتها: شهرهای مختلف**
  - **فعالیتها: حرکت بین شهرها**
- **جستجو: دنباله ای از شهرها مثل:آراد، سیبیو، فاگارس، بخارست**
  - **این جستجو با توجه به کم هزینه ترین مسیر انتخاب میشود**

مسأله :

حالت اولیه: حالتی که عامل از آن شروع میکند.

- **در مثال رومانی: شهر آراد n(Arad)**

تابع جانشین: توصیفی از فعالیتهای ممکن که برای عامل مهیا است.

- **در مثال رومانی: S(Arad)={ Zerind,Sibui,Timisoara}**

فضای حالت: مجموعه ای از حالتها که از حالت اولیه میتوان به آنها رسید.

- **در مثال رومانی: کلیه شهرها که با شروع از آراد میتوان به آنها رسید**

تابع جانشین + حالت اولیه = فضای حالت

آزمون هدف: تعیین میکند که آیا حالت خاصی، حالت هدف است یا خیر

- **هدف صریح: در مثال رومانی، رسیدن به بخارست**
- **هدف انتزاعی: در مثال شطرنج، رسیدن به حالت کیش و مات**

مسیر: دنباله ای از حالتها که دنباله ای از فعالیتها را به هم متصل میکند.

- **در مثال رومانی: Arad, Sibiu, Fagaras یک مسیر است**

هزینه مسیر: برای هر مسیر یک هزینه عددی در نظر میگیرد.

- **در مثال رومانی: طول مسیر بین شهرها بر حسب کیلومتر**

راه حل مسئله مسیری از حالت اولیه به حالت هدف است

راه حل بهینه کمترین هزینه مسیر را دارد

حال که اندکی با مبانی حل مسأله به روش جستجو آشنا شدید، در این جلسه سه مثال دیگر مطرح میکنیم و در جلسه آینده به "اندازه گیری کارایی حل مسأله" میپردازیم.

مثال: دنیای جارو برقی

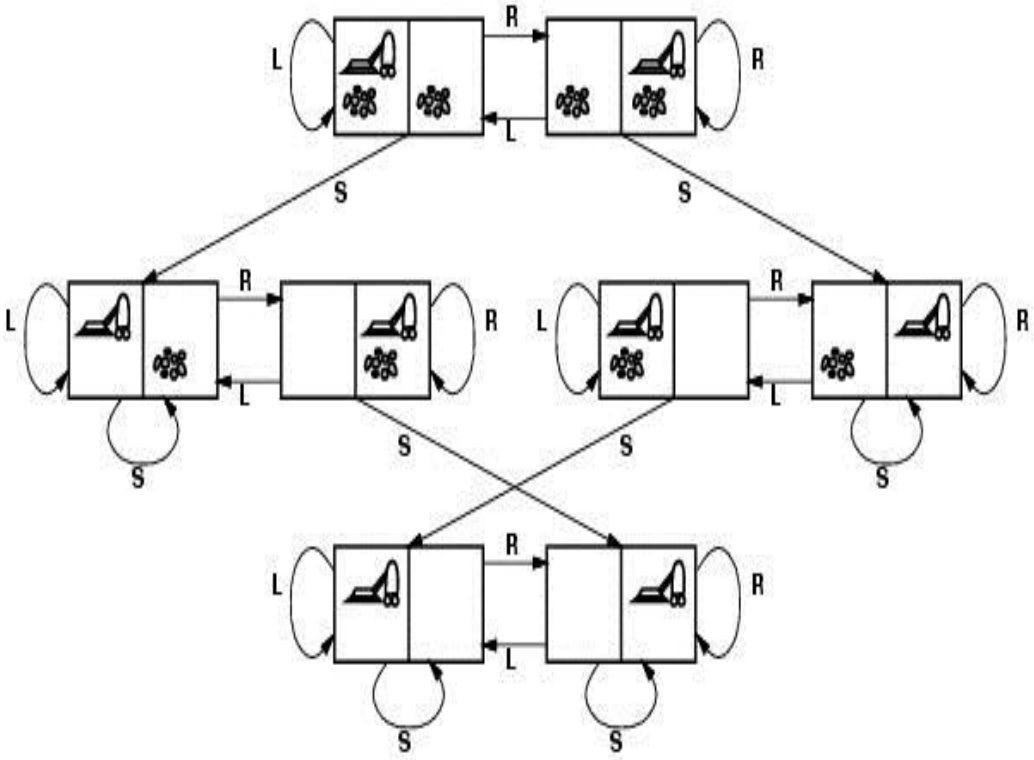
حالتها: دو مکان که هر یک ممکن است کثیف یا تمیز باشند.لذا ۸ = ۲*۲*۲حالت در این جهان وجود دارد

حالت اولیه: هر حالتی میتواند به عنوان حالت اولیه طراحی شود

تابع جانشین: حالتهای معتبر از سه عملیات: راست، چپ، مکش

آزمون هدف: تمیزی تمام مربعها

هزینه مسیر: تعداد مراحل در مسیر



مثال: معمای۸

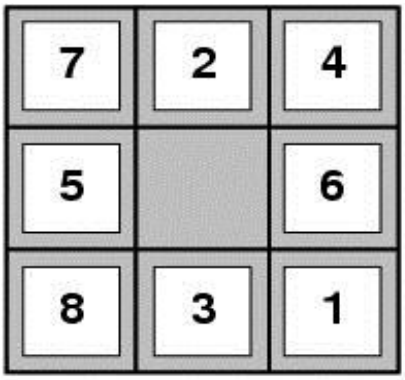
حالتها: مکان هر هشت خانه شماره دار و خانه خالی در یکی از ۹ خانه

حالت اولیه: هر حالتی را میتوان به عنوان حالت اولیه در نظر گرفت

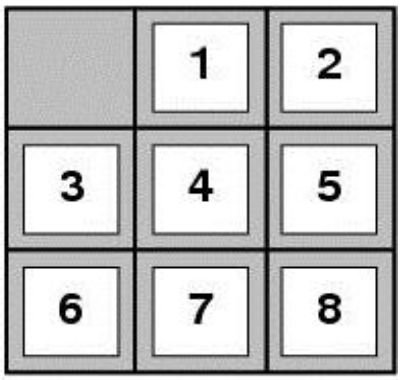
تابع جانشین: حالتهای معتبر از چهار عمل، انتقال خانه خالی به چپ، راست، بالا یا پایین

آزمون هدف: بررسی میکند که حالتی که اعداد به ترتیب چیده شده اند(طبق شکل روبرو) رخ داده است یا خیر

هزینه مسیر: برابر با تعداد مراحل در مسیر



Start State



Goal State

مثال: مسئله ۸ وزیر

فرمول بندی افزایشی

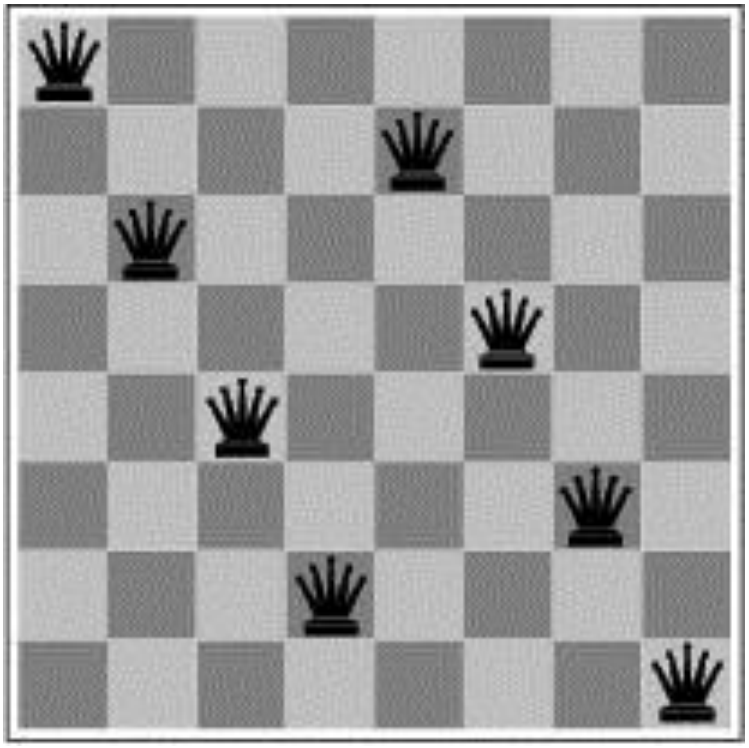
حالتها: هر ترتیبی از ۰ تا ۸ وزیر در صفحه، یک حالت است

حالت اولیه: هیچ وزیری در صفحه نیست

تابع جانشین: وزیری را به خانه خالی اضافه میکند

آزمون هدف: ۸وزیر در صفحه وجود دارند و هیچ کدام به یکدیگر گارد نمیگیرند

در این فرمول بندی باید ۱۴^۱۰*۳ دنباله ممکن بررسی میشود



فرمول بندی حالت کامل

حالتها: چیدمان n وزیر (۰≤ n≤ ۸)، بطوریکه در هر ستون از n ستون سمت چپ، یک وزیر قرار گیرد و هیچ دو وزیری بهم گارد نگیرند

حالت اولیه: با ۸ وزیر در صفحه شروع میشود

تابع جانشین: وزیری را در سمت چپ ترین ستون خالی قرار میدهد، بطوری که هیچ وزیری آن را گارد ندهد

آزمون هدف: ۸وزیر در صفحه وجود دارند و هیچ کدام به یکدیگر گارد نمیگیرند

این فرمول بندی فضای حالت را از ۳*۱۰^۱۴ به ۲۰۵۷ کاهش میدهد.

خواننده گرامی :

شاید با مطالعه این بخش، سوالی در ذهن شما بوجود آمده باشد که اصلا ارتباط این مباحث با شبیه سازی فوتبال دوبعدی چیست؟ در پاسخ به این سوال به موقع باید گفت شما برای طراحی الگوریتم های خود نیازمند افکار هوشمند و کاربردی هستید. شما در حال شبیه سازی یک روبات هستید و میخواهید تا میتوانید قدرت تصمیم گیری به روبات شبیه سازی شده تان بدهید و کاری کنید که روبات بدون استفاده از دستورات مستقیم و با هوش و ذکاوت خود کار کند. لازمه این کار داشتن آشنایی خوبی با دانش هوش مصنوعی (Artificial Intelligence) است. ما برای اینکه میخواهیم این علم را بصورت بنیادین و از ابتدا برایتان توصیف کنیم و با پیشرفت در این حوزه، استفاده هایی که میتوان از هوش مصنوعی در رشته های روبوکاپی کرد را بررسی کنیم، توصیه میکنیم با مطالعه کلیه جزوات ما و همچنین مطالعه کتب زبان اصلی و تخصصی در این حوزه، این دانش را بصورت حرفه ای فراگیرید.

به امید پیشرفت روبوکاپ

پایان قسمت هفتم

مرجع روبوکاپ ایران