دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

گزارش درس کارشناسی

هوش مصنوعی (۳)

پیاده‌سازی بازی تصادفی Monopoly با دو بازیکن

هستی برقراریان

استاد درس

مهدی قطعی

مدرس کارگاه

بهنام یوسفی‌مهر

فروردین ۱۴۰۲

چکیده

در این پژوهش، سعی داریم بازی Monopoly را با دو بازیکن و در زبان پایتون پیاده‌سازی کنیم.

واژه‌های کلیدی:

مونوپولی، تئوری بازی، الگوریتم مینیماکس، الگوریتم Expectminimax

فصل اول

مقدمه

مقدمه

بازی مونوپولی یکی از معروف‌ترین بازی‌های جذاب و استراتژیک است که در سراسر جهان محبوبیت فراوانی دارد. این بازی که در دهه ۱۹۳۰ میلادی توسعه داده شد، در اصل یک بازی مبادله، سرمایه‌گذاری و مدیریت منابع مالی است. هدف اصلی در مونوپولی، به دست آوردن ثروت و کنترل تمامی املاک و خانه‌های موجود بر روی تخته بازی است. در طول بازی، بازیکنان با خرید و فروش خانه‌ها، پرداخت مالیات، دریافت کارت‌های شانس و با مدیریت منابع مالی خود، سعی در برتری بر سایر بازیکنان دارند.

در این پروژه، هدف ما پیاده‌سازی یک بازی ساده مونوپولی با دو بازیکن است. با استفاده از زبان برنامه‌نویسی پایتون و قدرت هوش مصنوعی، تلاش می‌کنیم تا تجربه‌ای شبیه به بازی مونوپولی را برای دو بازیکن فراهم کنیم. در نسخه‌ی ساده‌ای که در این پروژه ارائه می‌شود، بازیکنان می‌توانند به ترتیب نوبت خود را پرتاب تاس کرده و موقعیت‌شان در تخته بازی را تغییر دهند. بازیکنی که بتواند املاک بیشتری را خریداری کند و به بهترین شکل از منابع مالی خود استفاده کند، به عنوان برنده بازی معرفی می‌شود.

برای پیاده‌سازی، از تکنیک‌های زیبای هوش مصنوعی استفاده می‌شود. به عنوان مثال، با استفاده از الگوریتم‌های تصادفی، پرتاب تاس را شبیه‌سازی می‌کنیم و با استفاده از الگوریتم‌های جستجو و تصمیم‌گیری، قوانین خرید و فروش املاک و مدیریت منابع مالی را پیاده‌سازی می‌کنیم. با این رویکرد، می‌توانیم یک محیط تعاملی و هوشمند را برای بازیکنان ایجاد کنیم که تصمیماتی هوشمندانه را در بازی انجام دهند.

در فصل دوم پیاده‌سازی را توضیح داده و در فصل سوم درباره‌ی یکی از الگوریتم‌هایی که می‌توانیم برای پیاده‌سازی به کار ببریم توضیح مختصری را ارائه می‌دهیم و در آخر، در فصل چهارم، جمع‌بندی و نتیجه‌گیری را داریم.

فصل دوم

پیاده‌سازی

پیاده‌سازی

وارد کردن کتابخانه‌های مورد نیاز (به عنوان مثال، کتابخانه random برای استفاده از توابع تصادفی):

```
import random
```

تعریف لیست players که شامل نام بازیکنان است:

```
players = ["Player 1", "Player 2"]
```

انتخاب یک بازیکن تصادفی به عنوان بازیکن فعلی:

```
current_player = random.choice(players)
```

تعریف لیست board که شامل نام خانه‌ها و اماکن مختلف تخته بازی مونوپولی است:

```
board = [
    "Go", "Mediterranean Avenue", "Community Chest", "Baltic Avenue",
    "Income Tax", "Reading Railroad", "Oriental Avenue", "Chance",
    "Vermont Avenue", "Connecticut Avenue", "Jail", "St. Charles Place",
    "Electric Company", "States Avenue", "Virginia Avenue", "Pennsylvania Railroad",
    "St. James Place", "Community Chest", "Tennessee Avenue", "New York Avenue",
    "Free Parking", "Kentucky Avenue", "Chance", "Indiana Avenue",
    "Illinois Avenue", "B&O Railroad", "Atlantic Avenue", "Ventnor Avenue",
    "Water Works", "Marvin Gardens", "Go To Jail", "Pacific Avenue",
```

```
    "North Carolina Avenue", "Community Chest", "Pennsylvania Avenue", "Short Line",
    "Chance", "Park Place", "Luxury Tax", "Boardwalk"
]
```

تعریف دیکشنری player_positions که نگه‌دارنده موقعیت هر بازیکن در تخته بازی است. این دیکشنری به ازای هر بازیکن مقدار اولیه صفر دارد:

```
player_positions = {player: 0 for player in players}
```

تعریف تابع roll_dice که یک عدد تصادفی بین ۱ تا ۶ را برمی‌گرداند (شبیه‌سازی پرتاب تاس):

```
def roll_dice():
    return random.randint(1, 6)
```

حلقه اصلی بازی:

چاپ نوبت بازیکن فعلی و خط ۲۰: انتظار برای فشار دادن Enter تا تاس پرتاب شود:

```
while True:
    print(f"It's {current_player}'s turn.")
    input("Press Enter to roll the dice...")
```

پرتاب تاس و نمایش نتیجه:

```
dice_roll = roll_dice()
    print(f"{current_player} rolled a {dice_roll}
```

افزایش موقعیت بازیکن فعلی با مقدار پرتاب تاس:

```
player_positions[current_player] += dice_roll
```

بررسی اینکه آیا بازیکن از خانه شروع عبور کرده است یا نه، و در صورت عبور، جابجایی موقعیت به صورت دوریکه و دریافت ۲۰۰ دلار:

```
if player_positions[current_player] >= len(board):
    player_positions[current_player] -= len(board)
    print(f"{current_player} passed Go and collected $200!")
```

چاپ مکان فعلی بازیکن فعلی بر روی تخته بازی:

```
  print(f"{current_player} landed
on{board[player_positions[current_player]]}.")
```

تغییر نوبت به بازیکن بعدی:

```
current_player = players[(players.index(current_player) + 1) % len(players)]
```

فصل سوم
الگوریتم Expectiminimax

الگوریتم Expectiminimax

الگوریتم expectiminimax یک الگوریتم تصمیم‌گیری است که در بازی‌های با حالت‌های احتمالی مورد استفاده قرار می‌گیرد. بازی مونوپولی نیز می‌تواند در برخی جنبه‌ها حالت‌های احتمالی داشته باشد، مانند پرتاب تاس یا کارت‌های شانس که نتایج تصادفی دارند.

در بازی مونوپولی هر بازیکن برای انتخاب حرکت بعدی خود، باید از وضعیت‌های فعلی بازی، گزینه‌های ممکن را در نظر بگیرد و بهترین انتخاب را انجام دهد. این انتخاب باید با هدف به دست آوردن نتیجه‌ای مثبت، مثلا بهبود وضعیت مالی خود یا افزایش احتمال برد صورت گیرد.

الگوریتم expectimax در واقع یک ترکیب از expectimax و minimax است. وقتی در بازی مونوپولی با حالت‌های احتمالی سر و کار داریم، می‌توانیم با استفاده از این الگوریتم، تصمیم‌های هوشمندانه‌تری برای بازیکنان ارائه دهیم.

عملکرد الگوریتم expectiminimax به این صورت است:

۱. برای هر حالت ممکن در بازی، یک ارزش تخمینی را محاسبه می‌کند. این ارزش می‌تواند معیاری از سود یا ضرر باشد.

۲. برای بازیکنان حریف، حالت‌های احتمالی را در نظر می‌گیرد و ارزش تخمینی برای هر حالت را محاسبه می‌کند. این ارزش تخمینی معیاری از امیدریاضی می‌باشد.

۳. بازیکن، حرکتی را انجام می‌دهد که منجر به حالتی شود که ارزش تخمینی آن بیشینه شود. این بازیکن به عنوان حریف در نظر گرفته شده است و تلاش می‌کند حرکتی انجام دهد که بهترین حالت ممکن برای بازیکن دیگر باشد.

۴. بازیکن ما، به دنبال حالتی است که ارزش تخمینی آن کمینه شود. این ارزش تخمینی نشان‌دهنده‌ی بدترین حالت ممکن برای خود است.

فصل چهارم

جمع‌بندی و نتیجه‌گیری

در این پروژه، ما به پیاده‌سازی بازی مونوپولی با دو بازیکن در زبان برنامه‌نویسی پایتون پرداختیم. با استفاده از توابع و ساختارهای داده مختلف، قوانین و قواعد بازی را پیاده‌سازی کردیم و یک محیط تعاملی برای بازیکنان فراهم کردیم.

ما از الگوریتم expectiminimax استفاده نکردیم و به جای آن، از الگوریتم‌ها و تکنیک‌های ساده‌تری برای تصمیم‌گیری و حرکت بازیکنان استفاده کردیم. با این حال، می‌توان با استفاده از الگوریتم expectiminimax بهبودی در استراتژی بازیکنان ایجاد کرد.

منابع و مراجع

[1] https://www.geeksforgeeks.org/expectimax-algorithm-in-game-theory/

[2] https://en.wikipedia.org/wiki/Monopoly_(game)

Abstract

In this research, we are trying to implement the Monopoly game with two players in Python.

Key Words: Monopoly, game theory, minimax algorithm, Expectminimax algorithm

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Mathematics and Computer science

AI project 3

Implementation of the random Monopoly game with two players

By
Hasti Barghararіyan

Supervisor
Dr. Mahdi Ghatee

Advisor
Behnam Yousefimehr

February 2023