• چکیده

مونوپولی یکی از بازی‌هایی است که به قوانین پیچیده‌ی آن معروف است. این بازی به صورت دو یا بیش از دو نفره روی یک صفحه با چهل خانه انجام می‌شود. استراتژی‌ها و تحقیقات مختلفی برای این بازی ارائه شده است که به بررسی چند مورد از معروف‌ترین‌های آن‌ها می‌پردازیم. همچنین شبیه‌سازی خود را از این بازی انجام داده و در نهایت با ایده گرفتن از استراتژی های ارائه شده و همچنین الگوریتم مینیمکس یک استراتژی برای شبیه‌سازی خود ارائه می‌دهیم.

واژه‌های کلیدی:

نظریه بازی، مینیمکس، مونوپولی

فهرست مطالب

صفحه

فهرست اشکال

صفحه

۱

فصل اول: مقدمه

مونوپولی یکی از بازیهای تخته‌ای[1] اســت که یه صــورت دو تا چند نفره انجام می‌شــود. هر بازیکن یک توکن[2] انتخاب کرده و در هر نوبت دو تاس انداخته و با توجه به اعداد ظاهر شده روی تاس‌ها، توکن خود را روی تخته حرکت می‌دهد. آنگاه با توجه به خانه‌ای که روی آن قرار می‌گیرد، باید یا با تصــمیم خود و یا به اجبار عملی را انجام دهد.

در بازی مونوپولی هر بایکن زمین‌هایی به دســت آورده و در صــورت امکان روی آن زمین‌ها خانه و هتل می‌ســازد. خرید خانه و ارتقای خانه‌ها به هتل باعث افزایش اجاره‌ای می‌شــود که بازیکنان دیگر در هنگام فرود آمدن روی خانه‌ی مورد نظر باید به حریفشان پرداخت کنند. با آنکه خرید و یا ارتقای خانه‌ها به هتل از مقدار پول یک بازیکن می‌کاهد، اما باعث می‌شــود به مجموع کل دارایی‌های او افزوده شــده و همچنین مقدار میانگین اجاره‌ای که از خانه‌های این دارایی‌ها دریافت می‌کند نیز بیشتر شود.

شبیه سازی این بازی با ساده سازی‌هایی صورت گرفته، ولی همچنان تلاش بر آن بوده که فا صله‌ی زیادی از خود بازی اصلی گرفته نشود. همچنین پایان بازی نیز به گونه‌ای تغییر داده شده که بازی خیلی طولانی نشــود، چراکه نشــان داده شــده با دو بازیکن و بدون معامله، به احتمال ۱۲٪ بازی پایانی ندارد (Friedman, 2009).

در نهایت اول یک عامل رندوم ارائه شــده که به صــورت رندوم تصــمیم‌گیری می‌کند. برای ایجاد یک عامل هوشمند برای این بازی، دو بخش برای استراتژی در نظر می‌گیریم. یکی از این استراتژی‌ها با کمک تحقیقات از قبل انجام شده و دیگری با ایده از الگوریتم مینیمکس ارائه شده است.

[1] Board games

[2] token

۲ فصل دوم: پیاده‌سازی بازی مونوپولی

در این فصل نحوه‌ی پیاده‌سازی بازی را به همراه قوانینی که در نظر گرفته شده تشریح می‌کنیم. این بازی در طی چند هفته و با مجموع بیش از ۴۰ کامیت در یک مخزن خصوصی گیت‌هاب پیاده‌سازی شد که کدهای آن ضمیمه شده است. همچنین در ابتدای پیاده سازی از مخزن (Wszdexdrf, بدون تاریخ) نیز استفاده شده است.

۲-۱-محدودیت‌ها

- بازی با دو بازیکن انجام می‌شود.
- بانک و شخصی که مدیریت آن را انجام دهد وجود ندارد، ولی همچنان بازیکنان به صورت غیرمستقیم با بانک ارتباط برقرار می‌کنند.
- هر بازیکن با یک نام منحصر به فرد از باقی بازیکن‌ها جدا می‌شود و لزومی به انتخاب یک شیء(مانند سگ، گربه، ماشین،...) نیست.
- کارت‌های سند مالکیت[۱] به طور مستقیم پیاده‌سازی نشده‌اند، اما اطلاعات آنها در کلاس هر دارایی ذخیره شده است.
- مزایده و معامله پیاده‌سازی نشده است.
- ساختن خانه، تخریب خانه پس از انداختن تاس و حرکت بازیکن و در واقع در انتهای نوبت او انجام می‌شود.
- برخلاف بازی فیزیکی که در کل ۳۲ خانه و ۱۲ هتل موجود است، در این شبیه‌سازی این محدودیت‌ها وجود ندارند.
- خانه‌های شانس[۲] و صندوق اعانه[۳] پیاده‌سازی نشده‌اند.
- کارت‌ها پیاده‌سازی نشده‌اند.
- ورشکست شدن به دو صورت اتمام کل پول بازیکن و یا ارزش کل[۴] وی در نظر گرفته می‌شود. عامل‌های بازی با هر دو تعریف امتحان شده‌اند.

---

۱ title deed cards

۲ Chance

۳ Community Chest

۴ net worth

- بازی به تعداد دفعات محدودی انجام می‌شود.

۲-۲-قوانین

- به هر بازیکن در ابتدای بازی ۵۰۰ واحد پول تعلق می‌گیرد.
- شروع بازی از خانه‌ی ۰(یعنی GO) و به صورت ساعتگرد دور تخته(شکل ۲-۱) انجام می‌گیرد.
- بازیکن شروع‌کننده‌ی بازی به صورت رندوم انتخاب می‌شود.
- بازی به صورت ترتیبی انجام شده و در هر نوبت، بازیکن مورد نظر با انداختن دو تاس به اندازه‌ی جمع مقدار ظاهر شده روی دو تاس جابه‌جا می‌شود.
- چنانچه تاس‌ها جفت بیایند، همان بازیکن دوباره تاس انداخته و بازی می‌کند. اما در صورتی که سه بار جفت بیاید، بازیکن باید به زندان منتقل شود.
- تخته از خانه‌های گوناگونی تشکیل شده که اکثر این خانه‌ها دارایی[۱] هستند. یک نوع از دارایی‌ها خیابان[۲] است که هرکدام رنگ مشخصی دارد. همچنین چهار نوع دارایی از نوع راه‌آهن[۳] و دو نوع دارایی شرکت خدماتی[۴] هستند.
- اگر بازیکن روی یک خانه‌ی دارایی فرود بیاید، در صورتی که آن خانه قبلاً خریداری نداشته باشد می‌تواند آن را خریداری کند. اما اگر حریف آن را خریداری کرده باشد، وی باید به حریف اجاره پرداخت کند.
- مقدار اجاره برای هر دارایی رنگی مشخص است و چنانچه تمام دارایی‌های یک رنگ مورد نظر خریداری شده باشد، مقدار آن دو برابر می‌شود. اگر خانه یا هتلی روی دارایی‌ها ساخته شده باشد، مقدار اجاره متفاوت خواهد بود که در پیاده‌سازی بر طبق (Wikibooks, 2022) لحاظ شده است.
- ساختن خانه روی خیابان‌های یک رنگ مشخص تنها در صورتی امکان‌پذیر است که بازیکن همه‌ی خیابان‌های آن رنگ مشخص را خریداری کرده باشد. البته این در صورتی است که هیچ

---

۱ property

۲ street

۳ railroad

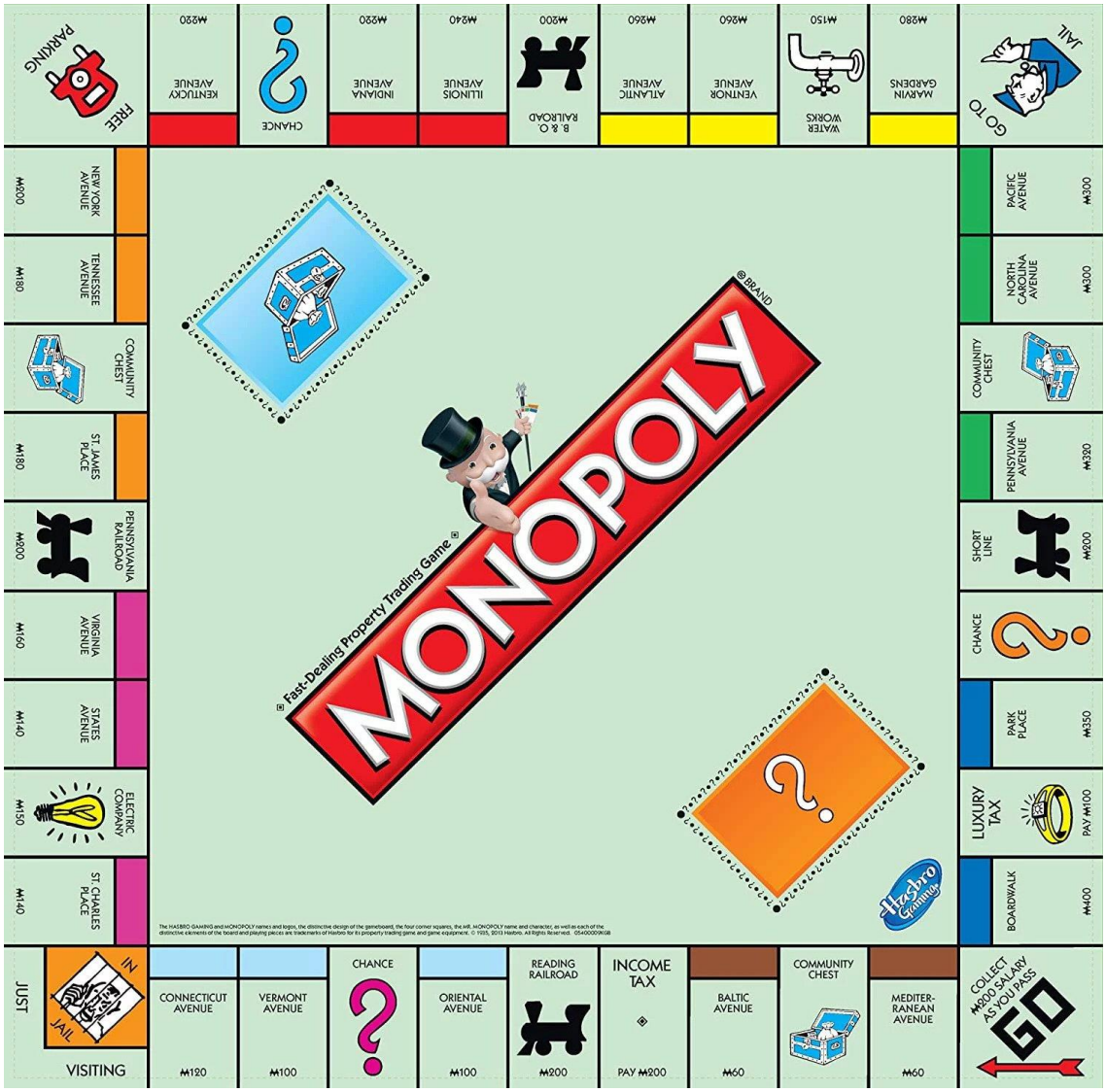
۴ utility

کدام از خانه‌های آن مجموعه رنگ مشخص رهن[1] داده نشد باشند. همچنین هزینه‌ی خانه‌ها برای هر خیابان نیز مطابق ((J.), 2023) گزیده شده است. همچنین هر خیابان حداکثر یک خانه بیشتر از سایر خیابان‌های مجموعه رنگی می‌تواند داشته باشد و بیشینه تعداد خانه‌های یک خیابان چهار است.

- در هر خیابانی که چهار خانه داشته باشد می‌توان چهار خانه را به یک هتل ارتقاء داد. هزینه‌ی این کار همان چهار خانه به علاوه‌ی قیمت هتل است که قیمت هتل با قیمت خانه در هر خیابان برابر است. بیشینه تعداد هتل‌ها در یک خیابان یک عدد است.
- اجاره‌ی هر کدام از راه‌آهن‌ها و شرکت‌های خدماتی مطابق (Wikibooks, 2022) تعیین می‌شود.
- فرود آمدن روی دو خانه‌ی ۴ و ۳۸ بازیکن را ملزم به پرداخت اجاره‌ای معین می‌کند.
- فرود آمدن روی خانه‌های شانس و صندوق اعانه و خانه‌ی ۲۰ اثری ندارد.
- در صورت فرود آمدن یا عبور از خانه‌ی ۰(GO)، بازیکن ۲۰۰ واحد پول کسب می‌کند.
- در صورت فرود آمدن روی خانه‌ی ۱۰(زندان) اتفاق خاصی برای بازیکن نمی‌افتد. اما در صورت فرود آمدن روی خانه‌ی ۳۰ یا سه بار متوالی جفت آوردن تاس، بازیکن به خانه‌ی ۱۰ منتقل شده و در صورت عبور از خانه‌ی ۰ نمی‌تواند ۲۰۰ واحد پول کسب کند.
- اگر در ابتدای نوبت خود بازیکن در زندان باشد، آنگاه باید یا ۵۰ واحد پول پرداخت کرده و از زندان خارج شود و یا تاس انداخته و در صورت جفت آمدن از زندان بیرون بیاید. در صورت خروج از زندان با توجه به تاس حرکت کرده و به طور معمول بازی را ادامه می‌دهد. اگر بازیکن در زندان بماند نمی‌تواند روی تخته حرکت کند، ولی می‌تواند عملیات دیگر(خرید، رهن،...) را انجام دهد. در صورتی که بازیکن سه بار متوالی در زندان باشد، به ناچار با پرداخت ۵۰ واحد پول خارج می‌شود.
- ساختمان‌های ساخته شده روی دارایی‌ها می‌توانند به قیمت نصف پولی که برای آنها پرداخت شده فروخته شوند؛ اما همچنان باید طوری فروخته شوند که تعداد خانه‌های روی هر دارایی از یک مجموعه رنگ حداکثر یکی بیشتر از تعداد خانه‌های باقی دارایی‌های روی آن مجموعه رنگ باشد. در صورت فروش یک هتل، چهار خانه‌ی قبلی به بازیکن بازگردانده می‌شوند.

[1] mortgage

- یکی از راه‌های افزایش پول بازیکن قرار دادن دارایی در رهن است. این در صورتی امکان دارد که ساختمانی روی دارایی ساخته نشده باشد. زمانی که دارایی در رهن باشد، بازیکن نمی‌تواند از آن اجاره کسب کند. بازیکن می‌تواند به میزان هزینه‌ی دریافت کرده برای رهن به علاوه ده درصد آن دارایی را از رهن خارج کند.
- بازی زمانی تمام می‌شود که یکی از بازیکن‌ها ورشکست[1] شود.



شکل۲-۱ تخته‌ی بازی مونوپولی کلاسیک با نام‌های آمریکایی که نام‌های بازی پیاده‌سازی شده نیز همین نام‌ها هستند.

[1] bankrupt

۳

فصل سوم: عامل‌های بازی

عامل اولی که پیاده سازی شده عامل رندوم[1] ا ست که تمام عملیات‌های او به صورت رندوم انجام می شود. سپس یک عامل هوشمند طراحی شده که طرز کار او در مقابل عامل رندوم و همچنین بازیکن حقیقی بررسی می‌شود.

۳-۱- پیاده‌سازی بازیکن

درک نحوه‌ی پیاده‌سازی بازیکن برای درک رفتار و استراتژی عامل‌ها مهم است. در ابتدای نوبت هر بازیکن چنانچه وی در زندان باشد تصمیم‌گیری اولیه انجام شده که آیا از زندان خارج شود یا خیر. آنگاه تاس انداخته شده و ادامه‌ی بازی مطابق قوانین ذکر شده ادامه داده می شود. پس از این مراحل چنانچه بازیکن همچنان در زندان باشد، حرکتی صورت نگرفته ولی می‌تواند فقط یکی از عملیات زیر را انجام دهد:

۱. ساختن فقط یک خانه روی یکی از دارایی‌هایی که مجموعه رنگشان کامل است به طوری که قوانین را به هم نزند.
۲. فروختن فقط یک خانه روی یکی از دارایی‌هایی که مجموعه رنگشان کامل است به طوری که قوانین را به هم نزند.
۳. ارتقای چهار خانه فقط روی یکی از دارایی‌ها
۴. فروختن فقط یک هتل و بازپس‌گیری چهار خانه
۵. در رهن قرار دادن فقط یک دارایی در صورتی که قوانین را برهم نزند.
۶. از رهن خارج کردن فقط یک دارایی در صورتی که قوانین را برهم نزند.

اگر بازیکن رو تخته حرکت کرده باشد باید یک عملیات متناظر با آن نیز صورت بگیرد.

عامل رندوم به گونه‌ای پیاده‌سازی شده که همه‌ی تصمیمات فوق را به صورت رندوم انجام می‌دهد. برای عامل هوشمندی که در ادامه شرح داده می‌شود دو استراتژی تفکیک شده در نظر گرفته شده است.

[1] random agent

۲-۳-استراتژی خروج از زندان

نخستین تصمیم‌گیری عامل بازی تصمیم درباره‌ی خروج از زندان یا ماندن در آن است که البته واضح است تنها زمانی که عامل در زندان باشد این تصمیم‌گیری انجام می‌شود. همان طور که در (Collins, n.d.) آمده، در ابتدای بازی که زمین‌های زیادی برای خرید موجود هستند، در صورت به زندان رفتن بهتر است سریع از آن خارج شد تا زمین‌های بیشتری را خریداری کرد، اما در انتهای بازی ماندن در زندان می‌تواند به صرفه‌تر باشد چراکه زمین‌های بیشتری توسط حریف خریداری شده و در صورت حرکت احتمال پرداخت اجاره بیشتر است.

بنابراین برای این قسمت از استراتژی با الگو گرفتن از همین ایده و همچنین الگوریتم Expectimax دو حالت ماندن در زندان و خروج از آن بررسی می‌شود. اگر عامل در زندان ماندن را انتخاب کند به او امتیاز ۰ تعلق می‌گیرد. در غیر این صورت با توجه به اعداد ظاهر شده روی تاس باید به یکی از خانه‌های تخته حرکت کند. هر یک از این حالات بررسی شده و طبق یک هیوریستیک میانگین گرفته می‌شود. این هیوریستیک در صورتیکه عامل روی یکی از دارایی‌های حریف قرار گرفته باشد اجاره‌ی متناظر را حساب کرده و در صورت ورشکست شدن ∞- را برمی‌گرداند. در غیر این صورت ۱- برگردانده می‌شود چون در هر صورت مقداری اجاره پرداخت شده است. به وضوح مقدار اجاره‌ی پرداختی نسبت به دارایی فعلی عامل در هر یک از خانه‌ها متفاوت است، اما برای سادگی در اینجا امتیاز منفی برای هر بار پرداخت اجاره ۱- در نظر گرفته می‌شود. حال اگر عامل روی خانه‌ای قرار بگیرد که بتواند آن را خریداری کند، فرض می‌کنیم حتماً آن را خریداری کرده و بنابراین در صورت ورشکست شدن عامل، هیوریستیک باز هم ∞- را برمی‌گرداند. اما اگر ورشکست نشود چون به ارزش کل عامل مقداری افزوده شده است، امتیاز ۱+ را کسب می‌کند. باز هم بهتر است که این امتیاز به صورت نسبی محاسبه شود اما برای سادگی و در این شبیه سازی همین امتیاز ۱+ هم کفایت می‌کند.

۳-۳-استراتژی کلی عامل

برای پیاده‌سازی استراتژی کلی عامل از الگوریتم Expectiminimax استفاده شده است. الگوریتم به این صورت است که اگر در گره‌ی بیشینه ساز[۱] قرار داشته باشد با توجه به اعداد روی دو تاس ۳۶ حالت برای آن وجود دارد. پس در واقع وارد هر یک از این گره‌های شانس[۲] شده و در آنجا نیز بسته به عملیات حریف وارد یکی از گره‌های فرزند می‌شود. پس با شروع از گره‌ی بیشینه‌ساز، وارد گره‌های شانسی می‌شود که در واقع نماینده‌ی فرود آمدن روی هر یک از خانه‌های ممکن است. فرزندان هر کدام از این گره‌های شانس، گره‌های کمینه‌ساز[۳] هستند که فرزندان این گره‌ها مجدداً گره‌های شانس بوده و این چرخه به همین صورت ادامه پیدا می‌کند. گره‌های برگ[۴] گره‌هایی هستند که در صورت صفر بودن عمق یا ورشکست شدن هر یک از بازیکن‌ها به آنها و یا ورود هر یک از بازیکن‌ها به زندان به آنها می‌رسیم. امتیاز وضعیت عامل مطابق هیوریستیکی که در ادامه شرح داده می‌شود در این برگ‌ها به دست می‌آید.

یکی از نکاتی که باید به آن اشاره کرد این است که در صورت جفت آوردن، الگوریتم بالا در حالات خاصی مطابق قوانین مشروح بازی عمل نمی‌کند. با این وجود تصمیم بر این شد که قوانین بازی تغییر داده نشوند و از ساده سازی‌های بیشتر خودداری شود؛ چون اولاً عامل در عمل با همین الگوریتم هم به خوبی عمل می‌کرد و ثانیاً این حالات در مقابل سایر حالات احتمال وقوع کمتری دارند.

هیوریستیکی که در این قسمت استفاده شده با بهره گرفتن از (Khan, 2018) به دست آمده است. این هیوریستیک به صورت زیر است:

---

۱ maximizer node

۲ chance nodes

۳ minimizer nodes

۴ leaf nodes

$$score = player's\ net\ worth - adversary's\ net\ worth + \sum_{player's\ properties} player's\ rent\ multiplier \times property's\ rent - \sum_{adversary's\ properties} adversary's\ rent\ multiplier \times property's\ rent$$

به علاوه اینکه یک مقدار $reserve$ هم تعریف می‌کنیم که در صورتی که مقدار پول عامل از این مقدار کمتر شود، یک جریمه به اندازه‌ی $reserve\ penalty$ به او تعلق می‌گیرد. می‌توان ضرایب دیگری برای ارزش کل بازیکنان نیز در نظر گرفت و یا تابع را گسترش داد که ما در اینجا به همین تابع بسنده کرده‌ایم.

۴

فصل چهارم: جمع‌بندی و نتیجه‌گیری

استراتژی کلی عامل با عمق ۴ و با ضرایب ۴۰۰، ۲، ۳ و ۵ به ترتیب برای $reserve$، $reserve\ penalty$، $player's\ rent\ multiplier$ و $adversary's\ rent\ multiplier$ اجرا شد. چیزی که مشاهده شد این بود که در عمق‌های کم تغییرات جزئی در ضرایب تاثیر چندانی نداشتند. عامل مورد نظر با توجه به مقدار پول کمی که در ابتدای بازی در اختیار داشت و در نظر گرفتن گام‌ها در کوتاه مدت تمایل زیادی به در رهن قرار دادن دارایی‌ها داشت که البته برای این شبیه‌سازی ساده منطقی بود. برخلاف عامل رندوم، عامل هو شمند طراحی شده را نتوانستیم به تعداد دفعات زیاد امتحان کرده و آمار دقیقی از عملکردش به دست آوردیم چون با رو به جلو رفتن بازی، سرعت عامل کمتر می‌شد. اما با تعداد دفعات کم و همچنین با بازی یک فرد حقیقی با عامل متوجه هوشمندی آن شده و به همین نتایج و استنباط‌ها بسنده کردیم.

۵ منابع و مراجع

(J.), J. (2023, March 17). *Buying Houses in Monopoly: Rules You Need to Know*. Retrieved from Monopoly Land: https://www.monopolyland.com/buying-houses-rules/

Collins, T. (n.d.). *Probabilities in the Game of Monopoly®*. Retrieved from Tkcs-Collins Website: http://www.tkcs-collins.com/truman/monopoly/monopoly.shtml

Friedman, E. J. (2009). Estimating the probability that the game of Monopoly never ends. *Proceedings of the 2009 Winter Simulation Conference (WSC)*, (pp. 380-391). doi:10.1109/WSC.2009.5429349

Khan, M. A. (2018). *AI for Board Games.* Retrieved from https://info.bb-ai.net/student_projects/project_reports/Games/Marya-Khan-Monopoly-project-report.pdf

Wikibooks. (2022). *Monopoly/Properties reference*. Retrieved from Wikibooks: https://en.wikibooks.org/w/index.php?title=Monopoly/Properties_reference&oldid=4219992

Wszdexdrf. (n.d.). *Monopoly: A simple Monopoly game implemented in Python*. Retrieved from Github: https://github.com/wszdexdrf/Monopoly