Dominika Wyszyńska AiR1, nr albumu: 318409

Projekt, który wykonałam to gra w pokera w odmianie Texas Holdem. Wybrana wersja nie ma ustalonego limitu podbijanej kwoty. Biorą w niej udział co najmniej dwie osoby, a maksymalnie dziewięć. W każdej rundzie ustalana jest kolejność graczy. Następnie wybierany jest Big Blind oraz Small Blind, którzy na starcie muszą wyłożyć pewną kwotę. W Texas Holdem mamy cztery części licytacji, podczas których gracze decydują, czy pasują, czekają lub podbijają stawkę. Możliwa jest również opcja va banque, w której jeden gracz podbija stawkę do wartości wszystkich swoich pieniędzy, a reszta może dorównać lub spasować. Pierwsza część licytacji polega na rozdaniu po 2 karty z talii dla każdego uczestnika, a następnie każdy z nich decyduje o dalszej grze. Drugą część (tzw. Flop) rozpoczyna wyłożenie 3 kart na stół, a następnie dochodzi do decydowania graczy. Trzecia część, czyli Turn, to wyłożenie 1 karty na stół i kolejne decyzje uczestników. Czwarta wygląda tak samo, jak trzecia i nazywa się River. Na koniec wyłaniani są zwycięzcy, a wygrana kwota jest przekazywana na ich konta. Poker polega na stworzeniu jak najsilniejszego pięciokartowego układu. Podczas gry mamy do dyspozycji 5 kart wspólnych na stole i 2 własne w dłoni.

Efektem projektu miała być gra jednoosobowa przeciwko wielu graczom komputerowym. Wykonane boty mają cechować się decyzjami zwiększającymi prawdopodobieństwo ich wygranej. Możliwa jest również rozgrywka z botem, który podejmuje losowe decyzje.

Projekt składa się z 10 plików. Cztery z nich to testy jednostkowe. Plik stale.py zawiera wartości stałe, które używane są do warunków podczas podejmowania mądrych decyzji przez bota.

Plik *karty.py* składa się z klasy *Karty* oraz dwóch słowników, z których tworzona jest cała talia kart. W konstruktorze klasy znajduje się lista kart gracza, a metoda *opis* wyczytuje jej zawartość. Metoda *rozdanie\_karty­* dodaje uczestnikowi określoną liczbę kart do układu, równocześnie przekładając je na koniec wspólnej talii. Reszta funkcji polega na sprawdzeniu kombinacji, które zachodzą w co najwyżej pięciu z posiadanych kartach. Kombinacje, jakie mogą wystąpić, to: poker królewski, poker, kareta, full, kolor, strit, trójka, 2 pary, para, wysoka karta.

Plik *metody.py* zawiera słownik oraz 2 metody. *Sortowanie\_ukladu\_kart* tworzy obiekt wysortowanych kart z układu oraz zwraca go z listą wysortowanych pozycji. Druga metoda (*mozliwe\_kombinacje*) sprawdza możliwe kombinacje, które mogą wystąpić w kartach uczestnika. Gdy dana kombinacja zajdzie, zapisywana jest wraz z jej punktacją, listą pozycji kart i uczestnikiem do słownika

Plik *uczestnik.py* zawiera 4 klasy: *Uczestnik*, *Gracz*, *Bot*, *Madry\_Bot*. Pierwsza z nich zawiera konstruktor z nazwą uczestnika, jego pieniędzmi i kartami. Pozostałe dziedziczą klasę *Uczestnik*. Posiadają też metody: *decyzja*, *dorównanie* i *obstawianie*. Klasa *Gracz* poświęcona została osobie, która uruchamia grę. Przy każdej decyzji wyświetlany jest w interfejsie spis opcji oraz zapytanie o wybór. Klasa *Bot* to bot, który wykonuje losowe ruchy podczas podejmowania decyzji. *Madry\_Bot* ma dodatkową metodę *sprawdzenie,* która rozpatruje wszystkie możliwe kombinacje w kartach uczestnika. Jeżeli karty mają małe szanse na jakąkolwiek kombinacje, bot pasuje. Gdy ma dużo pieniędzy lub dobre karty, podbija stawkę. W przypadku małej ilości pieniędzy lub wyjątkowo dobrych kart, gracz wybiera opcję va banque.

Plik *rozgrywka.py* zawiera klasę *Rozgrywka*. Konstruktor składa się z listy uczestników, gracza, listy botów i kart na stole. Słownik *gra* zawiera informacje o tym, czy uczestnik nadal bierze udział w rozgrywce oraz ile pieniędzy już wyłożył na stół.

Metody klasy *Rozgrywka*:   
- *rozdanie\_kart* – rozdaje po 2 karty z talii uczestnikom  
- *sprawdzenie\_kombinacji*- *sprawdzenie\_ukladu\_dla\_kart*  
- *zwyciezca* – wyłania listę zwycięzców spośród tych, którzy grają i mieli najlepsze kombinacje kart  
- *porównanie\_graczy* – znajduje wśród wygranej kombinacji graczy z najlepszymi kartami  
- *kolejna\_runda* – czyści globalne zmienne oraz tasuje karty przed nową rundą  
- *decyzje\_uczestnikow* – sprawdza decyzję każdego z pozostałych graczy w wybranej części rundy  
- *suma\_wygranej* – zwraca całkowitą sumę pieniędzy wyłożonych na stół podczas rundy  
- *rozdanie\_karty\_na\_stol*- *wrzucenie\_karty\_na\_koniec\_talii*  
- *nowe\_karty* – wykłada wybraną ilość kart na stół  
- *sprawdzenie\_stawki*  
- *opis\_kart\_na\_stole*  
- *czesc\_licytacji*, *flop*, *turn*, *river* – kolejne części rozgrywki pokera  
- *play* – główna część ustalająca kolejność wykonywania reszty metod klasy, schemat rozgrywki pokera   
- *zmiana\_kolejnosci\_graczy* – po rundzie wrzuca pierwszego gracza na koniec listy, aby role Big Blind oraz Small Blind się zmieniały

Za pomocą pliku *gra.py* należy uruchamiać grę. Najpierw tworzy się gracza, boty, a następnie wpisuje jako obiekt *Rozgrywki*. Wywołanie metody *play* rozpoczyna grę w pokera w konsoli. Gdy jest kolej gracza, należy wybrać decyzję z podanych opcji. Na koniec rozgrywki wyświetla się zwycięzca.

Jedną z trudniejszych części projektu było wymyślenie mądrego bota. Problemem było podjęcie decyzji, kiedy bot ma pasować lub podbijać. Sprawdzając kombinacje kart w rękach, można przypuszczać, czy jedna z nich zajdzie przy dalszym wykładaniu kart na stół. W ten sposób robot potrafi przewidywać sytuacje oraz podbijać, gdy ma dobre karty.

Kolejnym problemem było wymyślenie części porównywania kart graczy i wyłaniania zwycięzcy. W przypadku gdy kilku graczy ma tą samą kombinację, należy sprawdzić czyje pięć kart jest lepsze. Problem rozwiązałam poprzez stworzenie słownika, do którego wrzucałam nazwę kombinacji, uczestnika, wynik najwyższej karty z 5 kart oraz listę kart. Gdy kilku graczy miało tą samą kombinację, wybierałam najwyższy wynik i porównywałam karty uczestników.

Ostatnią przeszkodą było wykonanie interfejsu w jednej klasie. Klasa *Rozgrywka* posiadała główne części gry, jednak metody *Gracza* zadają pytania podczas podejmowania decyzji. Interfejs można byłoby wykonać tylko w *Rozgrywce*, jednak kod wyglądałby mniej czytelnie oraz metoda decyzji uczestników byłaby zbyt zagnieżdżona.

Wykonane dwa boty: mądry i randomowy. Ten drugi wykonuje losowe ruchy. Im bardziej głupi jest bot, tym bardziej jest skłonny do ryzyka bez żadnych kalkulacji. Gra z samymi głupimi botami jest trudna, gdyż rzadkim przypadkiem jest wytrwanie 2 uczestników do końca rundy. Często podbijają dużą kwotą lub grają va banque, co utrudnia rozgrywkę. Dodatkowo takie ryzykowne zagrania sprawiają, że uczestnicy szybko odpadają, a gra jest krótsza. Natomiast mądry bot podejmuje decyzje, wnioskując z kalkulacji. Ciężko przewidzieć, czy blefuje, więc gra jest ciekawsza i mniej ryzykowna.

Uważam, że moje rozwiązanie projektu jest bardzo dobre. Zamieszczone boty o różnym stopniu inteligencji, sprawiają, że gra nie jest jednorodna, nużąca. Odwzorowana jest gra poker Texas Holdem. Są tam zachowane wszystkie zasady takie jak wejście va banque, big blind, small blind.