Wydział Podstawowych Problemów Techniki
Politechnika Wrocławska

Blackjack

analiza strategii

Monika Tworek
Nr indeksu: 229776

Praca inżynierska napisana
pod kierunkiem
dr hab. Szymona Żeberskiego

Politechnika
Wrocławska
Wrocław 2018

Spis treści

Wstęp

„Ciekawość to pierwszy stopień do piekła lub pracy inżynierskiej”

Najbardziej widowiskowe użycie matematyki (a zwłaszcza statystyki) można znaleźć w kasynie. Z dostępnych gier największą popularnością cieszy się Blackjack ze względu na proste zasady. Praca zajmuje się analizą strategii, które można stosować w tej grze karcianej z elementami losowymi. W trakcie działania aplikacji można eksperymentalnie sprawdzic przewagę kasyna w poszczególnych, najczęściej spotykanych strategiach w kasynie.

Celem pracy jest:

- Zaprojektowanie i implementacja aplikacji, która będzie się składać z dwóch części:
  - ◇ Pierwsza, którą jest gra w Blackjacka, w której użytkownik gra przeciwko krupierowi
  - ◇ Druga, działająca podczas gry użytkownika, wylicza wyniki poszczególnych strategii na podstawie tych samych danych początkowych.
- Analiza danych dostarczonych przez aplikację.

Bardzo łatwo znaleźć aplikacje, które są tylko i wyłącznie grą, z różnie zaimplementowanymi odmiennymi zasadami, ilością graczy, czy nakierowane na konkretne urządzenie docelowe. Przeglądając artykuły można natrafić na „magiczne” zasady, którymi należy się kierować, aby zawsze wygrać, w której podawana jest często przewaga kasyna bez żadnych uzasadnień. Nie ma jednak możliwości sprawdzenia strategii indywidualnej gracza w stosunku do najczęściej stosowanych oraz praktycznego porównania wyników większej ilości danych.

Praca składa się z X rozdziałów:
Pierwszy rozdział opisuje zarysowany problem oraz zasady gry.
W drugim rozdziale znajdują się strategie wraz z algorytmami implementacyjnymi.
W rozdziale trzecim przedstawiono szczegółowy projekt w notacji UML, szkic programu oraz opis dokumentacji technicznej kodów źródłowych.
Czwarty rozdział poświęcony jest instalacji i uruchomieniu.
Piąty rozdział zajmuje się analizą otrzymanych eksperymentalnie wyników.

Analiza problemu

2.1 Blackjack-zasady

Aplikacja służy do analizy wyników otrzymanych poprzez stosowanie najczęstszych strategii używanych w kasynowej wersji „oczka". Blackjack to gra karciana, w której użytkownik mierzy się przeciwko krupierowi poprzez uzyskanie w wartości jak najbliżej (lecz nie więcej niż) 21. Punktacja kart jest następująca: figury są warte 10 punktów, as jest liczony za 1 lub 11 punktów (zależy od lepszego dla gracza wyniku), a pozostałe mają tyle punktów jaką mają wartość numerową. Gra rozpoczyna się od stawienia zakładu (w aplikacji wynosi ona 10). Następnie krupier otrzymuje jedną zakrytą, a drugą odkrytą kartę, a gracz dwie odkryte.
W początkowym momencie rundy gracz może:

- ubezpieczyć się (insure) - jeżeli widoczną kartą krupiera jest as, gracz może postawić dodatkową stawkę, że krupier ma blackjacka i ten zakład wygrywa się 2 do 1;
- rozdwoić karty (split) - jeżeli karty gracza są tej samej wartości, użytkownik dokłada dodatkową stawkę (10) i gra wtedy na dwie ręce;
- podwoić stawkę (double down) - gracz stawia dodatkową stawkę i otrzymuje tylko jedną kartę dodatkowo;

Ponadto użytkownik może dobrać kartę (hit) lub przestać dobierać (stand). Ręka przestaje być aktywna po wykonaniu polecenia stand i double down. W przypadku przekroczeniu 21 punktów ręka przestaje być aktywna i gracz automatycznie przegrywa, a runda nie jest rozwiązywana. Kiedy gracz nie może już dobierać kart, to runda jest rozwiązywana. Rozwiązanie rundy polega na odsłonięciu odwróconej karty krupiera, a następnie krupier dobiera karty tak długo jak ma mniej niż 17 punktów. Po przekroczeniu 17 punktów przestaje dobierać bez względu na ilość punktów u gracza.

Jeżeli gracz ma Blackjacka - dłoń wartości 21 przy tylko 2 kartach, to otrzymuje 2.5 stawki wygranej, w pozostałych wypadkach 2-krotną, w aplikacji to odpowiednio 25 dla blackjacka i 20 dla zwykłego zwycięstwa.

Aplikacja umożliwia grę tylko jednemu graczowi przeciwko krupierowi.

2.2 Strategie

Jest wiele strategii, które można podzielić na trzy kategorie, w której decyzja gracza jest uzależniona:

- od wcześniejszych decyzji i kart, które zostały użyte,
- tylko od ilości punktów, które ma na ręce,
- od innych czynników, często nieprzewidywalnych.

Program eksperymentalnie sprawdza jakie są wyniki poszczególnych strategii na tej samej talii kart, która została wcześniej przetasowana. Analizie poddano tylko grę jednego użytkownika bez sprawdzania wpływu na wynik użytkownika decyzji innych graczy tego samego stołu, którzy mają duży wpływ na wynik końcowy. Jedyny czynnik losowy to inna kolejność kart, gdyż krupier działa deterministycznie według strategii takiej jak w kasynach.

Ze względu na bardzo dużą złożoność obliczeniową algorytmu do obliczania strategii idealnej, powyżej 1 talii aplikacja sprawdza wszystkie strategie poza idealną.

Intuicja podpowiada, że część strategii nie sprawdza się dobrze jak na przykład zbyt mocne dążenie do 21 nie zwracając uwagi, że przekraczając tę wartość przegrywamy. Wiadomo, że bez znajomości kart jest wręcz niemożliwe każdorazowe wygrywanie gier. Znajomość jakie karty mogą się pojawić mogą za to zwiększyć szanse na wygrane. Jeżeli użytkownik się chwilę zastanowi jest w stanie stwierdzić, że strategia krupierska raczej będzie oscylować po ujemnej stronie wokół zera, bo kiedy gracz przekroczy 21 punktów nie są sprawdzane karty krupiera, który wygrał rundę.

Strategie

3.1 Idealna

Strategia ta jako jedyna zna układ potasowanej talii. Biorąc pod uwagę deterministyczną strategię krupiera, korzystając z nawrotów oraz odkładania na stos stanu stołu wraz z decyzją wybierana jest najlepsza procedura odpowiednich ciągnięć (hit) i pasów (stand).
W pierwszej iteracji wykonywane są same pasy aż skończą się karty. Przed każdym ruchem odkładany jest na stos stan stołu wraz z decyzją „Stand". Uzyskany wynik jest zapisywany jako idealny. Następnie wykonywany jest algorytm do momentu, gdy skończą się karty, a na stosie będą tylko stany stołu z decyzjami „Hit". Algorytm sprawdza, czy obecnie nie ma już kart. Wtedy należy porównać stan tablicy ze wynikiem idealnym. Jeżeli obecny wynik jest lepszy jest zapisywany jako idealny. Następnie ściągamy ze stosu wartości aż do momentu, gdy ściągnięta została tablica z decyzją „Stand". Wówczas odkładamy stan stołu z „Hit" na stos i dobieramy kartę. Jeżeli są jeszcze karty w talii, to odkładamy na stos stan stołu z „Stand" i wykonujemy pas. Ze względu na dużą złożoność obliczeniową w pracy została zaimplementowana bardziej optymalna wersja. Więcej o tym znajduje się w rozdziale XXX.

Pseudokod 3.1: Strategia Idealna

```
   Input: Tablica table
   Output: Wynik [win, draw, loos, blackjack, score]
 1 stos.append(table, "None")
 2 while table.deck has cards do
 3 |   table.begin_game()
 4 |   if table.player.hand is playing then
 5 |   |   stos.append(table, "Stand") table.stand()
 6 besttable ← table
 7 while table.deck has cards or stos has not only "Hit" do
 8 |   if table.deck hasn't cards then
 9 |   |   if table is better than besttable then
10 |   |   |   besttable ← table
11 |   |   table ← stos.pop()
12 |   |   while table.last_choice is "Hit" do
13 |   |   |   table ← stos.pop()
14 |   |   while table.player.hand isn't playing do
15 |   |   |   table.begin_game()
16 |   |   stos.append(table, "Hit")
17 |   |   table.hit()
18 |   else
19 |   |   while table.player.hand isn't playing do
20 |   |   |   table.begin_game()
21 |   |   stos.append(table, "Stand")
22 |   |   table.stand()
```

3.2 Krupierska

Jest to ta sama strategia według której gra krupier w kasynie. Należy ona do grupy strategii, w której gracz podejmuje decyzję uzależniając ją od wartości kart na ręce. Jeżeli „ręka" użytkownika ma wartość 16 punktów lub mniej w kartach należy wykonać polecenie Hit. W przypadku, czyli gdy gracz ma 17 oczek i więcej nie dobiera kart.

Pseudokod 3.2: Krupierska

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
while table.deck has cards do
    table.begin_game()
    while table.player.hand is playing and table.player.hand.value < 17 do
        table.hit()
    if table.player.hand is playing then
        table.stand()
```

3.3 Ekspansyjna

Strategia ta dąży do osiągnięcia jak najbliższej wartości 21. Również należy do grupy strategii uzależniającej decyzję gracza od aktualnie posiadanej liczby punktów na ręce. Gracz dobiera kartę dopóki nie ma co najmniej 20 oczek. Jeżeli ma 20 oczek pasuje. W przypadku oczka gra jest rozwiązywana.

Pseudokod 3.3: Ekspansyjna

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
while table.deck has cards do
    table.begin_game()
    while table.player.hand is playing and table.player.hand.value < 20 do
        table.hit()
    if table.player.hand is playing then
        table.stand()
```

3.4 Never bust

Jest bardzo podobna do strategii krupierskiej, czyli również decyzję użytkownika uzależnia tylko od ilości punktów na ręce. Gracz przestaje dobierać karty gdy ma więcej niż 11 punktów. Jeżeli ma 11 punktów lub mniej ciągnie kartę. Jest to strategia, która pozwala na przekroczenie 21, żeby każdorazowo krupier musiał grać.

Pseudokod 3.4: Never bust

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
while table.deck has cards do
    table.begin_game()
    while table.player.hand is playing and table.player.hand.value < 11 do
        table.hit()
    if table.player.hand is playing then
        table.stand()
```

3.5 Prawdopodobna

Ta strategia należy do strategii, które swoją decyzję uzależniają od wcześniejszych decyzji i kart, które zostały użyte. W tym wypadku ważne są użyte karty, które obniżają prawdopodobieństwo pojawienia się karty o wartości takiej jak wyciągnięta. Przed rozpoczęciem gry wyliczane jest prawdopodobieństwo wyciągnięcia kart dla każdej figury wiedząc, że w jednej talii znajdują się 4 takie karty. Następnie po każdej wyciągniętej karcie aktualizowane jest prawdopodobieństwo dla danej figury. W momencie, gdy po rozpoczęciu rundy gracz może ciągnąć lub pauzować, to sumowane jest prawdopodobieństwo wyciągnięcia karty, której ciągnięcie nie przekroczy 21. Jeżeli prawdopodobieństwo jest większe niż 50%, to gracz wykonuje „Hit", a w przeciwnym „Stand".

Pseudokod 3.5: Prawdopodobna

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
Actuall(probably)
while table.deck has cards do
    table.begin_game()
    Actuall(probably)
    while table.player.hand is playing and probably(21 − table.player.hand.value) > 50% do
        table.hit()
    if table.player.hand is playing then
        table.stand()
```

3.6 Filmowa inaczej Hi-Low

Najbardziej medialna i filmowa strategia, która polega na liczeniu oczek. Należy do strategii, które decyzję o ciągnięciu uzależniają od wcześniejszych decyzji i użytych kart. Tutaj tak jak w prawdopodobnej ważne są użyte karty, które zmieniają stan stołu. Niskie karty (od 2 do 6) mają wartość +1. Średnie karty (od 7 do 9) są neutralne i mają wartość 0. Pozostałe, wysokie karty mają wartość -1. Przed rozpoczęciem rozdania stół ma wartość 0. Wraz ze wzrostem wyniku rośnie prawdopodobieństwo, że w talii pozostaje więcej wysokich kart i odwrotnie - jeżeli potrzebna jest niska karta, a wynik jest ujemny, to oznacza, że jest wysokie prawdopodobieństwo karty z przedziału 2-6. Po każdej wyciągniętej karcie aktualizowany jest stan stołu. W momencie, gdy różnica między wartością ręki, a 21 wynosi więcej niż 10 oczek, a stan stołu jest dodatni, to gracz ciągnie kartę. Jeżeli różnica między 21, a ilością oczek na ręce użytkownika jest mniejsza od 11, ale większa od 8, a stan stołu jest w okolicy zera (-1, 0, 1), to użytkownik ciągnie kartę. Jeżeli wartości ręki gracza jest większa niż 13 (czyli różnica między 21, a ilością punktów gracza jest mniejsza od 8), a stan stołu jest ujemny, to gracz ciągnie kartę. W pozostałych wypadkach użytkownik pasuje.

Pseudokod 3.6: Hi-Low

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
while table.deck has cards do
    table.begin_game()
    Actual(state, table.croupier.hand)
    Actual(state, table.player.hand)
    while table.player.hand is playing do
        if 21 − table.player.hand.value > 10 and state > 0 then
            table.hit()
        if 21 − table.player.hand.value > 8 and statein[−1, 0, 1] then
            table.hit()
        if 21 − table.player.hand.value > 0 and state < 0 then
            table.hit()
        else
            table.stand()
```

3.7 Intuicyjna

Jest to strategia starająca się symulować jak najbardziej grę opierającą się na intuicyjnym podejściu do ciągnięcia lub pasowania. Należy ona do strategii, które decyzję podejmują w zależności od innych, w tym wypadku nieprzewidywalnych czynników. Często intuicyjne podszepty są sugerowane otoczeniem, usłyszanym ostatnio zdaniem bądź zapachem, który aktualnie się unosi. Nie jest możliwe odtworzenie takich warunków, więc w celu przybliżenia różnych czynników, które potrafią się szybko zmienić, losowana jest liczba z przedziału od 0 do 100. Jeżeli wylosowany wynik jest niższy niż 50, to traktuje się to jako podszept duszy mówiącej, by pasować. W przypadku wylosowania liczby większej bądź równej 50 gracz ciągnie kartę.

Pseudokod 3.7: Intuicyjna

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
while table.deck has cards do
    table.begin_game()
    number ← Random(0,100) while table.player.hand is playing and random > 50 do
        table.hit()
    if table.player.hand is playing then
        table.stand()
```

3.8 Reaguj na bank

Ta strategia należy do trzeciej grupy wymienionej w 2.2 Strategie i uzależnia decyzję gracza od widocznej karty krupiera. Jeżeli ma ona wartość 2 lub 3, to gracz pasuje, gdy posiada co najmniej 13 punktów, a ciągnie jeżeli wartość na ręce nie przekracza 13 oczek. Jeżeli karta krupiera jest z przedziału od 4 do 7, to gracz przestaje dobierać karty, gdy ma co najmniej 17 punktów, a w przypadku kiedy ma więcej niż 17, to pasuje. Dla tych kart krupiera strategia przypomina zasady strategii krupierskiej. W pozostałych przypadkach, czyli kiedy wartość widocznej karty krupiera jest większa niż 7 punktów, to gracz pasuje kiedy ma co najmniej 18 oczek. Kiedy ma mniej niż 18 punktów to ciągnie kartę.

Pseudokod 3.8: Reaguj na bank

Input: Tablica *table*
Output: Wynik $[win, draw, loos, blackjack, score]$

```
1  while table.deck has cards do
2      table.begin_game()
3      if table.croupier.hand.value in [2,3] then
4          while table.player.hand.value < 13 do
5              table.hit()
6      if table.croupier.hand.value in [4,5,6,7] then
7          while table.player.hand.value < 17 do
8              table.hit()
9      if table.croupier.hand.value > 7 then
10         while table.player.hand.value < 18 do
11             table.hit()
12     if table.player.hand is playing then
13         table.stand()
```

3.9 Podstawowa

Jest to strategia, która swoją decyzję uzależnia w pełni od wartości widocznej karty krupiera oraz wartości ręki gracza. Ponadto rozróżnia sytuację, gdy gracz rozpoczyna grę z parą lub gdy jedna z kart to as. Wszystkie zależności pokazuje poniższy rysunek.

Karty gracza	Karty krupiera									
	2	3	4	5	6	7	8	9	10	A
Punkty (bez par i asów)										
5,6,7,8	H	H	H	H	H	H	H	H	H	H
9	H	D	D	D	D	H	H	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
17+	S	S	S	S	S	S	S	S	S	S
Pary z asem										
A-2	H	H	H	D	D	H	H	H	H	H
A-3	H	H	H	D	D	H	H	H	H	H
A-4	H	H	D	D	D	H	H	H	H	H
A-5	H	H	D	D	D	H	H	H	H	H
A-6	H	D	D	D	D	H	H	H	H	H
A-7	S	D	D	D	D	S	S	H	H	H
A-8	S	S	S	S	S	S	S	S	S	S
A-9	S	S	S	S	S	S	S	S	S	S
Pary										
A-A	P	P	P	P	P	P	P	P	P	P
2-2	P	P	P	P	P	P	H	H	H	H
3-3	P	P	P	P	P	P	H	H	H	H
4-4	H	H	H	P	P	H	H	H	H	H
5-5	D	D	D	D	D	D	D	D	H	H
6-6	P	P	P	P	P	H	H	H	H	H
7-7	P	P	P	P	P	P	H	H	H	H
8-8	P	P	P	P	P	P	P	P	P	P
9-9	P	P	P	P	P	S	P	P	S	S
10-10	S	S	S	S	S	S	S	S	S	S

Legenda:

H	Hit - dobierz kartę
S	Stand - nie dobieraj kart
D	Double down - podwój stawkę
P	Split - rozdziel karty

Rysunek 3.1: Strategia podstawowa

3.10 Przełam passę

Strategia ta jest uzależniona od wcześniejszych decyzji gracza, a dokładniej jego wyników. Jest często spotykana jako odbicie się od dna przez zatwardziałych graczy, którzy często zostawiają duże pieniądze w kasynach. Na początku gracz kiedy ma mniej niż 15 punktów ciągnie kartę. W momencie, gdy wartość jego ręki przekroczy 15, a on wciąż może grać, to pasuje. Jednak kiedy miał co najmniej 3 porażki pod rząd, to

najmniejsza wartość ręki, która zmienia decyzję gracza z Hit na Stand wzrasta do 17 punktów. Jeżeli więc gracz miał powyżej 3 porażek pod rząd, to kiedy ma 16 oczek bądź mniej to ciągnie kolejną kartą, a gdy ma 17 punktów i więcej to pasuje. Jeżeli gracz miał co najmniej 5 porażek z rzędu, to pasuje dopiero kiedy ma 19 oczek lub więcej. Jeżeli wartość ręki ma poniżej tej liczby, to ciągnie kartę. Po każdym zwycięstwie licznik porażek jest zerowany.

Pseudokod 3.9: Przełam passę

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
state ← 0
while table.deck has cards do
    table.begin_game()
    if state < 3 then
        if table.player.hand.value < 15 then
            table.hit()
        else
            table.stand()
    if statein[3, 4] then
        if table.player.hand.value < 17 then
            table.hit()
        else
            table.stand()
    if state > 4 then
        if table.player.hand.value < 19 then
            table.hit()
        else
            table.stand()
    if table.winner is Croupier then
        state ← state + 1
    else
        state ← 0
```

3.11 Przetrzymaj passę

Strategia ta jest bardzo podobna do Przełam passę i również opiera się na wcześniejszych decyzjach gracza, a w tym przypadku wygranych. W przypadku tej strategii jednak jest obniżany próg od kiedy gracz zaczyna pasować, a nie podwyższany. Na początku gracz przestaje dobierać jak ma co najmniej 17 punktów, czyli zaczyna się tak jak w przypadku strategii krupierskiej. W przypadku co najmniej 3 porażek z rzędu, to użytkownik przestaje dobierać jeżeli ma 14 oczek lub więcej. Jeżeli ma co najwyżej 13 punktów, to ciągnie kartę. Jeżeli gracz ma co najmniej 5 porażek, to pasuje, gdy ma powyżej 10 punktów. Kiedy ma mniej, to ciągnie kartę. Po każdym zwycięstwie zerowany jest licznik porażek.

Pseudokod 3.10: Przetrzymaj passę

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
state ← 0 while table.deck has cards do
    table.begin_game()
    if state < 3 then
        if table.player.hand.value < 17 then
            table.hit()
        else
            table.stand()
    if statein[3, 4] then
        if table.player.hand.value < 14 then
            table.hit()
        else
            table.stand()
    if state > 4 then
        if table.player.hand.value < 11 then
            table.hit()
        else
            table.stand()
    if table.winner is Croupier then
        state ← state + 1
    else
        state ← 0
```

3.12 Zależna od szczęścia

Ta strategia przypomina strategie Przełam i Przetrzymaj passę. Na początku użytkownik przestaje ciągnąć karty jeżeli ma co najmniej 15 punktów. Gdy ma co najwyżej 14 oczek ciągnie kartę. Następnie licznik jest zwiększany z każdym zwycięstwem o 1 aż do 6, a z każdą porażką jest obniżany o 1 aż do -4. W zależności od stanu licznika zmienia się granica po której gracz przestaje ciągnąć karty. Wraz z każdym zwycięstwem jest zwiększana o 1 aż do 20, a z każdą porażką jest obniżana o 1 aż do 11 oczek na ręce gracza. Dobrze to obrazuje poniższa tabelka:

stan licznika	granica
-4	11
-3	12
-2	13
-1	14
0	15
1	16
2	17
3	18
4	19
5	20

Gdzie granica oznacza liczbę, która musi być większa od wartości ręki gracza, by ten mógł dalej ciągnąć kartę.

Pseudokod 3.11: Zależna od szczęścia

```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
state ← 0 depend ←
[−4 ← 11
 − 3 ← 12
 − 2 ← 13
 − 1 ← 14
 0 ← 15
 1 ← 16
 2 ← 17
 3 ← 18
 4 ← 19
 5 ← 20
 ]
while table.deck has cards do
    table.begin_game()
    if self.table.player.hand.value < depend[state then
        table.hit()
    else
        table.stand()
    if table.winner is Croupier then
        if state > −4 then
            state ← state − 1
    else
        if state < 6 then
            state ← state + 1
```

3.13 Pasująca

Jest to strategia, która zalicza się do strategii uzależnionych od innych, nieprzewidzianych czynników. W tej strategii gracz każdorazowo po rozpoczęciu gry pasuje bez względu na stan stołu, czy wartości ręki. W przypadku tej strategii liczy się, że będzie się miało więcej punktów niż krupier lub krupier podczas ciągnięcia kart przekroczy 21, co spowoduje wygraną gracza.

Pseudokod 3.12: Pasuja

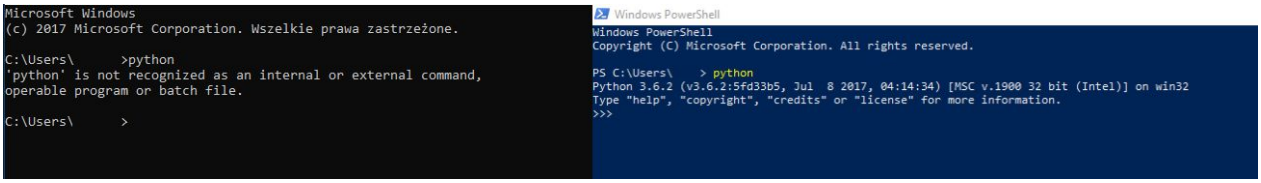
```
Input: Tablica table
Output: Wynik [win, draw, loos, blackjack, score]
while table.deck has cards do
    table.begin_game()
    if table.player.hand is playing then
        table.stand()
```

Projekt aplikacji

Implementacja i instalacja

5.1 Wymagania techniczne

Praca była pisana na systemie Windows 10 z wykorzystaniem Pythona 3.6.2 oraz Angulara 7.0.6 (Node 8.11.2), i korzystano z przeglądarki chrome. Ponieważ aplikacja została napisana za pomocą pythona wymagane jest, by użytkownik posiadał interpreter do Pythona 3.x.x. Najbezpieczniej jest pobrać ze strony autora [6]. W celu sprawdzenia, czy zainstalowano wystarczy w terminalu wpisać python. Po lewej stronie widać, że system nie rozpoznaje polecenia i należy go zainstalować:



Rysunek 5.1: Porównanie wyników terminali

Nie należy zapomnieć o dodaniu pythona do PATH - bez tego program nie działa poprawnie. Warto to zaznaczyć w instalatorze:

Rysunek 5.2: Instalator z zaznaczonym dodaniem Python do PATH

W programie użyto bibliotek, które wymagają dodatkowej instalacji - jsonscheme [8], flask [7] oraz namedlist [9]. Instaluje się je wpisując w terminal następujące polecenie:

```
$ pip install flask
```

```
$ pip install jsonscheme
$ pip install namedlist
```

Może się okazać, że należy zauktualizować pipa (poleceniem „python -m pip install –upgrade pip”).
W celu uruchomienia backendowego serwera należy w terminalu w folderze z Blackjack wywołać polecenie:

```
$ python blackjack.py
```

W tym momencie uruchomi się serwer, co zostanie wyświetlone na terminalu:

```
* Serving Flask app "blackjack.server" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://localhost:5000/ (Press CTRL+C to quit)
```

Rysunek 5.3: Widok terminala po uruchomieniu serwera pythonowego

Następnie należy zainstalować środowisko do Angulara [1] aby móc zobaczyć jak program działa w swojej przeglądarce. Aby móc zainstalować Angulara należy mieć zainstalowane następujące narzędzia: git [2], npm [4] i nodejs [3]. Czasmi by system je „zauważył” po instalacji należy ponownie uruchomić komputer.
W celu zainstalowania Angulara należy wykonać w treminalu polecenie:

```
$ npm install -g @angular/cli
```

Następnie w folderze angular-client w terminalu wywołać polecenie:

```
$ ng serve
```

Po uruchomieniu można zobaczyć, czy można już korzystać w przeglądarce z programu:

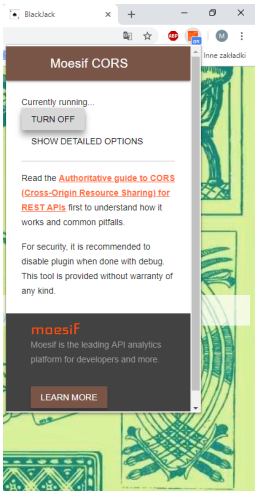
```
                        angular-client> ng serve
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Date: 2018-12-23T18:10:51.455Z
Hash: ab3a798a8c0c20d0b1ea
Time: 20086ms
chunk {main} main.js, main.js.map (main) 83.7 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 223 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.08 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 32 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.31 MB [initial] [rendered]
i ｢wdm｣: Compiled successfully.
```

Rysunek 5.4: Widok terminala po uruchomieniu serwera angularowego

Kiedy użytkownik uruchomi pythonowy serwer, uruchomi serwer angularowy i wejdzie na localhost:4200 w przeglądarce pojawi się strona startowa aplikacji:

Rysunek 5.5: Widok przeglądarki ze stroną startową aplikacji

Niestety ponieważ dane są na serwerze localhost:5000, a użytkowniko korzysta z portu 4200, to jest to sprzeczne z polityką CORS. Ten problem został omówiony w porpzednim rozdziale w XXX. Aby móc wygenerować dane lub rozpocząć grę należy dodać rozszerzenie do przeglądarki, które wyłączy politykę CORS. W przypadku przeglądarki chrome to rozszerzenia "Moesif Origin CORS Changer"w wersji 0.2.11:



Rysunek 5.6: Włączone rozszerzenie wyłączające politykę CORS

Analiza uzyskanych danych

6.1 Próbka badawcza

Dla 1 talii jest 52! różnych ułożeń kart. Jeżeli chcielibyśmy przybliżyć tę liczbę otrzymujemy $8.0658 * 10^{67}$. Jest to ogromna liczba - dla porównania wiek obserwowalnego wszechświata to „zaledwie" 10^{18} sekund. Jest więc niemożliwe, by dla 1 talii wyliczyć wynik wszystkich strategii dla każdego rozdania. W przypadku 2 talii ta liczba to $\frac{104!}{2^{52}}$, co daje ok. $2{,}3 * 10^{150}$, a przy 3 ($\frac{(52*3)!}{3!^{52}}$, czyli $2{,}5 * 10^{235}$) i 4 talii ($\frac{(52*4)!}{4!^{52}} \approx 4 * 10^{321}$) ciężko jest nawet znaleźć obrazowe przybliżenie wielkości tej liczby. W przypadku gry Blackjacka nie musimy rozróżniać, czy otrzymana karta o wartości 7 jest koloru kier, karo, trefl, czy pik liczba rozdań zdecydowanie się zmniejsza. Wtedy każda talia to nie 52 rozróżnialne karty, ale $\frac{(13*4)!}{4!^{13}}$ co daje nam „tylko" $9{,}2 * 10^{49}$. Ponieważ nie jesteśmy sprawdzić wszystkich możliwych rozdań zostało to sprawdzone dla XX gier z użyciem 1 talii, YY dla 2 talii oraz odpowiednio ZZ i AA dla 3 i 4 talii.

6.2 Podsumowanie

Ogólna przewaga kasyna w ilości zwycięstw gracza do jego porażek wacha się między 20% a 30%. W skrajnych przypadkach wynosi ona od 50% do nawet 70%. Jest część strategii, które mają większą liczbę zwycięstw niż porażek i wychodzą na plus - Przełam Passę i Podstawowa. Czasam zdarzają się ciekawe rozdania, gdy wszystkie strategie poza idealną wychodzą na minus lub gdy dla danego rozdania jakaś strategia daje ten sam wynik co idealna (w czasie zbierania danych wydarzyło się tak dla PrzełamPassę). Co ciekawe największą liczbę blackjacków miała strategia Intuicyjna. Czasami były rozdania, gdy tylko jedna strategia, która w ogólnym rozrachunku jest na dużym minusie była jedyną strategią (poza idealną) wychodzącą na plus po zakończeniu gry.
Można zauważyć, że zdecydowanie lepsze są strategie bardziej defensywne (Pasująca, NeverBust) lub opierająca się na prawdopodobieństwie wyciągnięcia odpowiedniej karty (Podstawowa, Prawdopodobna). Są strategie, które mają podobne wyniki co Intuicyjna (Ekspansyjna), co wskazuje na to, że ich zasady nie są dopasowane do BlackJacka.
Warto zwrócić uwagę na to, że jeżeli przekroczymy 21 oczek, to krupier nie musi grać. Gdyby ta zasada została zmieniona, to wiele porażek zamieniłoby się w remis, bo wartości ręki krupiera często by przekraczała 21 oczek. Jest to jeden z argumentów dlaczego kasynom opłaca się mieć tę prostą grę w swojej ofercie.

6.3 Porównanie wyników na wykresach

Tutaj są wykresy i odpowiedni komentarz do nich. Wykres dla 1 talii. Wykres dla 2 talii. Wykres dla 3 talii. Wykres dla 4 talii.

6.4 Idealna

W przypadku rozgrywek dla jednej talii strategia ta osiągała niemal brak przegranych, a wynik różnił się tylko ilością blackjacków dla danego rozdaniu. Wraz ze zwiększeniem ilości talli zmniejszyła się też skuteczność

gry, co pokazuje, że często są rozdania, gdy nawet pomimo znania układu talii gracz nie jest w stanie całkowicie ograć kasyna. Można też zauważyć, że często opłaca się raz przegrać rundę, by potem móc wygrać 7 rund z rzędu niż starać się zwyciężać każdą rundę.

6.5 Krupierska

Pomimo stosowania tej samej strategii co krupier często wychodzi się przy tej strategii na minus. Pokazuje to, że 17 jest zbyt wysoką granicą dla bezpiecznej gry przeciwko kasynu.

6.6 Ekspansyjna

Ta strategia uzyskała najgorszy wynik z wszystkich analizowanych. Jednocześnie jest jedną z najczęściej stosowanych strategii przez początkowych graczy, dzięki czemu kasyna wygrywają ogromne kwoty. W przypadku nieco ponad 300 rund statystycznie gracz traci około 1400 złotych. Jest to jednocześnie strategia wygrywająca najmniej rund poprzez blackjacka.

6.7 Never bust

Strategia ta bardzo podobna do strategii krupierskiej, ale mająca niższą granicę po której gracz pasuje wbrew pozorom osiąga lepsze wyniki. Nie są one drastycznie lepsze (różnica około 2%), ale pokazuje, że gdy gracz waha się między wyższym bądź niższym progiem do pasowania warto wybrać niższy.

6.8 Prawdopodobna

Znajduje się ona w czołówce trzech najlepszy strategii bez znajomości układu kart obok Podstawowej i Przełam Passę. Zarówno ta jak i Podstawowa strategia bazują na prawdopodobieństwu nie przekroczenia 21 jednak Podstawowa zachowuje pewien margines, co pozwala jej uzyskać lepsze wyniki. W tej strategii staramy się jak najbezpieczniej zbliżyć się do „oczka” bez uwzględnienia strategii krupiera.

6.9 Filmowa inaczej Hi-Low

Najbardziej widowiskowa strategia wbrew pozorom nie osiąga najlepszych wyników. Statystycznie wygrywa się co 3 rundę dzięki tej strategii, co w ogólnym rozrachunku wychodzi na minus. Ma jednak zdecydowaną przewagę dla gracza - jest najszybsza do zapamiętania i wdrożenia oraz pomaga ćwiczyć pamięć i podstawowe operacje matematyczne.

6.10 Intuicyjna

Ze względu na swój nieprzewidywalny charakter pokazuje, że pod względem matematycznym intuicyjne „Teraz pasuję” daje niemal najgorszy wynik ogólny. Gorsze wyniki daje tylko strategia Ekspansyjna. Warto jednak zauważyć pewną ciekawą zależność - mimo stosunkowo niewielkiej ilości wygranych, to w tej strategii najczęściej pada blackjack. Nie tylko porównując, że gdy statycznie na 1600 zwycięstw pada nieco ponad 160 blackjacków (bez uwzględnienia strategii idealnej to około 150 na 1400), to przy tej strategi średnio na 4,6 zwycięstw pada 1 blackjack, co jest zadziwiającym wynikiem.

6.11 Reaguj na bank

Strategia ta ma jeden z wyższych wyników, a jednocześnie nie jest trudna do zapamiętania, co powoduje, że często pojawia się przy poruszaniu tematu strategii blackjackowych. Mimo dobrej proporcji liczby zwycięstw do porażek niestety przy tej strategii wychodzi się na minusie.

6.12 Podstawowa

Jest to jedna z nielicznych strategii, która pozwala na wychodzenie na plus jednak ma bardzo dużo skomplikowanych zależności i wymaga nauczenia się wielu warunków. Jako jedna z nielicznych wykorzystuje split w przypadku gry uczestnika.

6.13 Przełam passę

Jedna z niewielu strategii wychodząca na plus. Patrząc po wynikach pozostałych strategii można odnieść wrażenie, że ponieważ nie miała zbyt wielu porażek, to granica 15 punktów jest jedną z bardziej optymalnych by przestać ciągnąć kartę. Strategia wychodziła na plus też ze względu na dość dużą ilość blackjacków. Po iluś porażkach często polegających na przekroczeniu 21 ciężej przekroczyć 21 i większa szansa na pokonanie krupiera.

6.14 Przetrzymaj passę

Mimo obniżania progu wynik jest ujemny, ale może być to kwestia mniejszej ilości dodatkowo punktowanych blackjacków. Przy jednej talii, gdzie średnia liczba gier to 9, może być ciężko uzyskać 5 porażek pod rząd stosując się do nie przekraczania ustalonego progu, który nie jest zbyt wysoki.

6.15 Zależna od szczęścia

Strategia ta jest niewiele poniżej zera jeżeli chodzi o wygrane pieniądze. Niestety wraz ze wzrostem liczby talii użytych w grze obniża się skuteczność tej strategii.

6.16 Pasująca

Mimo iż logicznie patrząc, że w sytuacji, gdy na ręce mamy 6 oczek i możemy bez obaw ciągnąć dalej, to w tej strategii mimo wszystko pasujemy pokazuje jak często kasyno może przegrać poprzez przekroczenie 21 punktów, gdyż ogólnie na ok. 400 rund zwycięskich jest ponad 160.

6.17 Przykładowe rozdanie

Poniżej znajduje się przykładowy układ kart dla 1 talii oraz jakie rozwiązanie jest idealne. Dla porównania wyniki pozostałych strategii.

6.18 Blackjacki

Poniżej znajdują się porównania, która strategia ma największą szansą na blackjacka

6.19 Przewaga bez blackjacków

Poniżej znajduje się porównanie jak zmienia się ilość pieniędzy jeżeli nie bierzemy pod uwagę faktu, że blackjack jest punktowany dodatkowo.

6.20 Porażki poprzez przekroczeni 21

Na ile patrząc na ile rund w czasie talii średnio się tak przegrywa. Jak często wtedy by przegrał krupier?? -¿ Dlaczego często kasyno wychodzi na plus

Na koniec cytat Steinhousa ”Między duchem, a materią pośredniczy matematyka

Podsumowanie

Bibliografia

[1] Angular technology. Web pages: https://cli.angular.io/.

[2] git technology. Web pages: https://git-scm.com/.

[3] nodejs enviromental. Web pages: https://nodejs.org/en/.

[4] npm enviromental. Web pages: https://www.npmjs.com/get-npm.

[5] Python documentation. Web pages: https://docs.python.org/3/library/.

[6] Python enviromental. Web pages: https://www.python.org/downloads/.

[7] Python flask library. Web pages: https://pypi.org/project/Flask/.

[8] Python jsoncheme library. Web pages: https://pypi.org/project/jsonschema/.

[9] Python namedlist library. Web pages: https://pypi.org/project/namedlist/.

[10] C. Drösser. *Matematyka daj się uwieść*. Wydawnictwo Naukowe PWN, year=2011.

Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.