FAKULTA ústav
STROJNÍHO automatizace
INŽENÝRSTVÍ a informatiky

Algoritmy umělé inteligence [VAI]

Jméno:	Skupina:	Datum zadání:
Patrik Lindner	4pAIŘ/1	–

Akademický rok:	Ročník:	Semestr:	Datum odevzdání:
2023/24	4	letní	2024

Vyučující:	Název úlohy:
Ing. Petr Šoustek	**Blackjack AI**

Blackjack

Blackjack je karetní hra odlišná především tím, že není založena pouze na náhodě, ale umožňuje pomocí různých strategií zvýšit pravděpodobnost výhry (především metoda tzv. počítání karet, kdy se hráč pokouší „zapamatovat“ tažené karty, má velkou publicitu).

Princip hry

Každý hráč na začátku hry obdrží dvě karty. Pak mu krupiér (dealer) nabízí další karty. Hráč se rozhoduje, zda bude chtít další kartu, nebo ne. Základní princip hry je, že hráč chce mít hodnotu karet blíže 21 než krupiér, ale přitom 21 nepřekročit. Hráč, který má v ruce součet karet větší než 21 automaticky prohrál, i když má nižší součet než krupiér (dealer). Karty od 2 do 10 mají při počítání stejnou hodnotu, jaká je uvedena na kartě. Janek, dáma a král mají hodnotu 10. Eso se může počítat za 1 nebo za 11, dle potřeby hráče.

Základním cílem je mít více bodů než krupiér (dealer). Pokud má hráč i krupiér stejný počet bodů, končí hra nerozhodně. Každý hráč hraje samostatně proti krupiérovi, je tedy možné, aby v průběhu jedné hry krupiér s různými hráči vyhrál, prohrál či remizoval.

Rozdávání karet

Karty rozdává krupiér (dealer) ve směru hodinových ručiček. Nejprve všem dá jednu kartu a pak druhou. Karty hráčů jsou všechny lícem vzhůru, kdežto krupiérova druhá karta je lícem dolů.

Možnosti hry

Možnosti hráčů jsou v této verzi hry omezeny a mohou volit pouze hit (vzít si další kartu) anebo stand (nevzít si již další kartu, čímž hráč ukončí svou hru). Poté co odehrají svou hru všichni hráči je na řadě krupiér (dealer). Ten nejprve otočí svou druhou katru lícem vzhůru a podle součtu hodnot karet provádí jeden ze dvou tahů podle následujícího pravidla. Pokud je součet karet menší nežli 17 krupiér musí vzít další kartu. Pokud součet karet přesáhl 17 krupiér už si nesmí vzít další kartu.

Pokud krupiér překročí 21 vyhráli všichni, kdo 21 nepřekročili. Pokud překročil 21 i hráč i krupiér, pak vyhrál krupiér. Pokud má hráč i krupiér stejný součet jedná se o remízu nikdo nepřijde o peníze. Jinak vyhrává ten kdo nepřekročil 21 a je k 21 blíž.

Vlastní implementace hry

Na začátku hry je vytvořen balíček karet, ten je poté zamíchán. Balíček je reprezentován seznamem hodnot karet.

```
deck = [2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11] * 4
#print(deck)
random.shuffle(deck)
```

Obrázek 1: Vytvoření balíčku karet a zamíchání

Poté jsou hráčům rozdány karty podle pravidel hry ve dvou kolech po jedné kartě v jistém pořadí.

```
card=deck.pop()
base_player_hand = [card]
```

Obrázek 2: rozdání první karty prvnímu hráči

```
card=deck.pop()
base_player_hand.append(card)
```

Obrázek 3: rozdání druhé karty prvnímu hráči

Po rozdání úvodních karet dealerem jsou karty zobrazeny tak, jakoby je hráči viděli ve skutečné hře.

```
#####    TABLE    #####

base_player_hand: [3, 2]
deck_counter_player_hand: [5, 10]
risk_deck_counter_player_hand: [6, 8]
player_hand: [3, 10]
memory_palace_player_hand: [10, 8]
dealer_hand: [ 4 , X]
```

Obrázek 4: Zobrazení stavu stolu po rozdání počátečních karet

Hráči

Do hry bylo implementováno několik hráčů, kde každý z nich využívá jinou strategii.

První hráč (Base Player) má strategii založenou na tom, jaké je viditelná dealerova karta. Tato strategie je převzatá z wikipedie a upravená tak aby hráč pouze vzal kartu anebo stál.

Vaše karty	Krupiérova karta									
	2	3	4	5	6	7	8	9	10	A
Tvrdý součet										
17-20	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	SU	SU	SU
15	S	S	S	S	S	H	H	H	SU	H
13-14	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
5-8	H	H	H	H	H	H	H	H	H	H

Obrázek 5: Strategie z wikipedie

```
#best move based on basic strategy (wiki)
def basic_strategy(base_player_hand, dealer_card):
    player_score = hand_value(base_player_hand)

    if player_score >= 17:
        return 's'  # stand/stay
    elif player_score <= 11:
        return 'h'  # hit/play
    else:
        return 's' if dealer_card in [2, 3, 4, 5, 6] else 'h'
```

Obrázek 6: implementace strategie z wikipedie v kódu

Druhý hráč (Deck Counter safe Player) používá klasickou strategii počítání karet na stole, přičemž pakliže jsou na stole vysoké karty s hodnotou 10 a 11 je jich v balíčku méně a je bezpečné z balíčku brát. Pakliže jsou na stole nízké karty není bezpečné odebírat karty z balíčku. Hráč tedy spočítá, kolik je na stole vysokých (10, 11) a nízkých (2, 3, 4, 5, 6) karet a podle toho kterých je více se poté rozhodne. Tato strategie samo o sobě není zcela vhodná, protože kdyby hráč dostal na začátku karty dávající v součtu méně než 12 nemá smysl zůstávat stát. Proto při těchto součtech byla strategie upravena, aby hráč v takových případech bral kartu vždy. Naopak při příliš vysokém součtu není vhodné brát kartu ani, když je to dle počítání karet bezpečné. Pouze počítání karet na stole nezohledňuje karty, které má sám hráč v ruce.

```python
#player who counts cards and uses max       safe
def count_cards_minimax(deck):
    # strategy/heuristics
    count = sum(1 if card in [2, 3, 4, 5, 6] else -1 if card in [10, 11] else 0 for card in deck)
    count += count #safe zone
    return 'h' if count < 0 else 's'
```

Obrázek 7: základní strategie počítání karet

```python
if hand_value(deck_counter_player_hand) < 18:
    if hand_value(deck_counter_player_hand) < 12:
        action = 'h'
    else:
        action = count_cards_minimax(table)
```

Obrázek 8: Úprava základní strategie počítání karet

Třetí hráč (Deck Counter risk Player) počítá karty v balíčku na stole na základě jeho momentálního součtu karet. Pokud se mu karta hodí odečte 1 pokud se mu karta nehodí v mysli přičte 1. Podle toho, zda je na konci v plusu nebo v mínusu se rozhodne, zda si vezme další kartu. Dále počítá s tím, že krupiérova karta, kterou nevidí, je ze seznamu karet pro něj vhodných. Takže ji pak ještě přičte ke konečnému součtu.

```python
def risk_count_cards_minimax(deck,risk_deck_counter_player_hand):
    #print(risk_deck_counter_player_hand)
    risk_sum = hand_value(risk_deck_counter_player_hand)
    risk_limit = 21-risk_sum
    good_list = [i for i in range(2, risk_limit+1)]     #bez es
    if hand_value(risk_deck_counter_player_hand)<21:
        if len(good_list)<10:
            good_list.append(11)        #s esy
    if len(good_list)>1:
        var_list = [good_list[-1]]
    else:
        var_list = [-1]
    # strategy/heuristics
    #count = sum(-1 if card in good_list else +1 if card in var_l
    count = sum(-1 if card in good_list else +1 for card in deck)
    count += 1  #scared part
    #print(count)
    return 'h' if count < 0 else 's'
```

Obrázek 9: Počítání karet v balíčku

Čtvrtým hráčem je sám uživatel (Player (you)). Jeho hra je automaticky ukončena, pokud přesáhne součet 20.

Všichni předešlí hráči hráli fér, bez podvádění, tak jak by to teoreticky mohl dokázat i člověk pokud by uměl dobře a rychle počítat. Následující hráči ale již nebudou tak čestní a budou používat techniky, které v kasinech nevidí rádi.

Pátý hráč (Memory palace Player) ví, jaké jsou karty v balíčku (buď používá paměťový palác nebo podvádí) a bere si tak, aby byl k hodnotě 21 co nejblíže, ale nikdy ji nepřesáhl. To znamená že může prohrát jen když dealer bude blíže k 21.

```
return 'h' if my_sum + deck[-1] <= 21 else 's'
```

Obrázek 10: Strategie hráče s paměťovým palácem

Šestým, pro uživatele volitelným, hráčem je tajný hráč kasina (Casino player). Ten stejně, jako pátý hráč podvádí, ale na rozdíl od pátého hráče nemaximalizuje svůj vlastní zisk, ale snaží se k výhře přiblížit dealera. Jako první tento hráč zjistí, zda si kasino (dealer) vůbec může brát karty. Poté zjistí, kolik si může vzít karet, pokud si tedy vůbec může nějakou vzít. Pak pro všechny možnosti podle počtu karet, které si může vzít, než přesáhne 21 a automaticky prohraje (už si dál nemůže brát karty) zjistí která z nich posune kasino (dealera) co nejblíže k 21 a tu zvolí. Pokud se hráč kasina nebo dealer dostanou do takové situace, že si nemůžou brát žádné karty nebo žádná z možností neposune dealera blíže k vítězství, pak tento hráč hraje tak, aby sám vyhrál, a tak nevypadal podezřele.

```
my_sum = hand_value(player_hand)
deal_sum = hand_value(dealer_hand)
if is_there_casino_player == True:
    if deal_sum<17:                                          #deal bude muset brat
        #jak dostat casino co nejblize 21
        if hand_value(player_hand)<21:                                  #paklize si vbc muze cas
            result=can_casino_win(deck,player_hand,dealer_hand,beat_hands)

            if result[0]==True:                                      #casino can win
                return result[1]        #vrati pocet kolik karet bych si mel vzit aby casino vyhralo
```

Obrázek 11: Kontrola, zda může hráč kasinu pomoci

```
#predpoklad ze jsem schopen vzit alespon jednu kartu je splnen vzdy kdyz
for element in range(0,how_many+1):   #budu potrebovat vsechny moznosti
    pom_pom_deck = pom_deck.copy()
    dealer_hand_var = dealer_hand.copy()
    poss_max = 0
    while (hand_value(dealer_hand_var+[(pom_pom_deck[-1])]))<=21:
        if hand_value(dealer_hand_var)<17:
            dealer_hand_var.append(pom_pom_deck.pop())
            poss_max=hand_value(dealer_hand_var)                    #ma
        else:
            break
    pom_deck.pop()
    poss.append(poss_max)

indx_max_val = poss.index(max(poss))
max_val = max(poss)

num_of_beat = 0
for i in beat_hands:
    if hand_value(i)<=21:
        if max_val>hand_value(i):
            num_of_beat += 1
    #else podle me nema smysl psat

if num_of_beat>1:
    return [True, indx_max_val]
else:
    return  [False, None]
```

Obrázek 12: Kontrola všech možností pomoci a výběr té nejlepší pakliže ani tak nikoho kasino (dealer) neporazí hraje hráč sám za sebe

Poslední hraje kasino (dealer) ten, jak již bylo řečeno, se řídí jednoduchým pravidlem. Pokud je jeho aktuální součet karet pod 17 bere si další kartu a jinak stojí.

```
#Dealer's turn
while hand_value(dealer_hand) < 17:
    dealer_hand.append(deck.pop())
```

Obrázek 13: Dealerova povinná strategie

Během celé hry zobrazuje program tahy jednotlivých hráčů stejně, jako by tomu bylo při skutečné hře. Poté, co všichni odehrají své hry program vyhodnotí, který hráč s kasinem vyhrál/remizoval/prohrál.

```
def win_check(player_score, dealer_score):
    if player_score>21:
        return "Lost (over 21)."
    elif dealer_score > 21 or player_score > dealer_score:
        return "Wins!"
    elif player_score == dealer_score:
        return "Draw!"
    else:
        return "Lost (Dealer was closer to 21)."
```

Obrázek 14: Určení vítězů a poražených

Na konci každé hry program zobrazí karty všech hráčů a zda vyhráli, či prohráli.

```
final hands and values


Base Player's Hand:              [6, 9, 10]  Value of the hand is:  25
Deck Counter safe Player's Hand: [10, 10]  Value of the hand is:  20
Deck Counter risk Player's Hand: [3, 10, 5]  Value of the hand is:  18
Players (your) Hand:             [11, 9]  Value of the hand is:  20
Memory palace Player's Hand:     [5, 10, 5]  Value of the hand is:  20
Casino Player's Hand:            [10, 10]  Value of the hand is:  20
Dealer's Hand:                   [11, 5, 3]  Value of the hand is:  19

final results

Base Player:                Lost (over 21).
Deck Counter safe Player:   Wins!
Deck Counter risk Player:   Lost (Dealer was closer to 21).
Player (you):               Wins!
Memory palace Player:       Wins!
Casino Player:              Wins!
```

Obrázek 15: Výsledky hry

Po každé odehrané hře je uživatel dotázán, zda chce hrát další hru, ve které si na začátku opět může zvolit, zda bude hrát i s hráčem kasina. Pakliže hráč již hrát nechce hra se ukončí a zobrazí se statistiky jednotlivých her.

```
[[15, 29, 6], [16, 33, 1], [15, 30, 5], [14, 31, 5], [17, 24, 9], [14, 32, 4]]
Stats for games with Casino Player
Base Player stats (Win/Loss/Draw):                [15, 29, 6]
Deck Counter safe Player stats (Win/Loss/Draw):   [16, 33, 1]
Deck Counter risk Player stats (Win/Loss/Draw):   [15, 30, 5]
Player (you) stats (Win/Loss/Draw):               [14, 31, 5]
Memory palace Player stats (Win/Loss/Draw):       [17, 24, 9]
Casino Player stats (Win/Loss/Draw):              [14, 32, 4]
[[15, 29, 6], [16, 28, 6], [20, 23, 7], [22, 26, 2], [28, 20, 2]]
Stats for games without Casino Player
Base Player stats (Win/Loss/Draw):                [15, 29, 6]
Deck Counter safe Player stats (Win/Loss/Draw):   [16, 28, 6]
Deck Counter risk Player stats (Win/Loss/Draw):   [20, 23, 7]
Player (you) stats (Win/Loss/Draw):               [22, 26, 2]
Memory palace Player stats (Win/Loss/Draw):       [28, 20, 2]
```

Obrázek 16: Výsledná statistika po 50 hrách s a bez hráče kasina

Jak je z předešlé statistiky vidět účast hráče kasina nejspíše téměř neovlivní výsledný výkon prvního hráče. Druhému hráči klesl počet remíz a zvýšil se počet proher. Tito dva hráči však moc neriskují, a tak ani žádné velké výkony nepodávají. U hráčů tři, čtyři a pět je již ztráta při zapojení hráče kasina znatelná.
Bez hráče kasina dealer prohraje z padesáti her s pěti hráči tedy z 250 možných výsledků 101krát. S hráčem kasina, jehož skóre se nepočítá, jelikož hraje s penězi kasina, dealer prohraje z 250 možných výsledků 77krát. Rozdíl těchto situací je asi 10 procent, což se zdá poměrně málo, ale musíme si uvědomit, že ve spoustě situací nemůže hráč kasina vůbec hru ovlivnit. Poté už by bylo na úvaze kasina, zda by se mu vyplatilo platit tohoto hráče nebo vyplácet výhry ostatních hráčů

Zdroje

Wikipedia. Online. Wikipedia. 2024. Dostupné z: https://cs.wikipedia.org/wiki/Blackjack. [cit. 2024-05-04].