7. Metoda Monte Carlo

Zadání:

Metoda Monte Carlo představuje rodinu metod a filosofický přístup k modelování jevů, který využívá vzorkování prostoru (například prostor čísel na herní kostce, které mohou padnout) pomocí pseudonáhodného generátoru čísel. Jelikož se jedná spíše o filosofii řešení problému, tak využití je téměř neomezené. Na hodinách jste viděli několik aplikací (optimalizace portfolia aktiv, řešení Monty Hall problému, integrace funkce, aj.). Nalezněte nějaký zajímavý problém, který nebyl na hodině řešen, a získejte o jeho řešení informace pomocí metody Monte Carlo. Můžete využít kódy ze sešitu z hodin, ale kontext úlohy se musí lišit.

Řešení:

Problém, který jsem se pokusil vyřešit metodou Monte Carlo spočíval v myšlence, jestli se mi vyplatí sázet do rulety. Vybral jsem si Evropskou verzi, která obsahuje pouze 1x nulu. Pravidla rulety jsou jednoduché. Hráč nebo hráči vsadí libovolný počet žetonů. Podle výběru sázení se při výhře znásobí vložený počet žetonů a znásobí se podle daných pravidel. Je mnoho možností sázek a mezi ně patří sázky na barvy, čísla, sudé, liché atd. Sázky se dají kombinovat. Mezi vybrané sázecí strategie jsem vybral sázky na:

1. Červenou (2x násobič)
2. Černou (2x násobič)
3. Zelenou (35x násobič)

Obrázek 1 Evropská ruleta

Simulaci jsem vytvořil jen za pomocí knihovny matplotlib (a coloramy na barvy konzole) kterou jsem využil k vytvoření grafů na vyhodnocení strategií. V simulaci jsem si vytvořil 2 třídy. Z nich vytvořené objekty využívám jako přepravky pro data, a to pro uložení dat o číslech a jejich výherní násobek a také pro uložení dat pro pozdější vložení do grafů a vyhodnocení strategie.

```
10  class cisloSazky:
11      vyherniCislo : int
12      multiplier : float
13      def __init__(self,vyherniCislo,multiplier) -> None:
14          self.vyherniCislo = vyherniCislo
15          self.multiplier = multiplier
16
17  class cisloGrafu:
18      i : int
19      vydelanePenize : int
20      pocetHitu : int
21      sazky : int
22      def __init__(self,i,vydelanePenize,pocetHitu,sazky) -> None:
23              self.i = i
24              self.vydelanePenize = vydelanePenize
25              self.pocetHitu = pocetHitu
26              self.sazky = sazky
```

Obrázek 2 Třídy na čísla do grafu a sázek

Zde mám jen definice na „obarvení“ čísel. Pod tím se nachází funkce která vezme tyto seznamy a vytvoří z nich seznam objektů typu cisloSazky a přidá jim správně násobič výhry. Tyto seznamy se později využijí v mé metodě Monte Carlo

```
29  cervena = [1,3,5,7,9,12,14,16,18,19,21,23,25,27,30,32,34,36]
30  cerna = [2, 4, 6, 8, 10, 11, 13, 15, 17, 20, 22, 24, 26, 28, 29, 31, 33, 35]
31  zelena = [0]
32
33  def pridejMultiplierKCislum(staryList : list, multiplier : float):
34      newList = list()
35
36      for vyherniCislo in staryList:
37          newList.append(cisloSazky(vyherniCislo, multiplier))
38
39      return newList
40
41
42  nasobnaZelena = pridejMultiplierKCislum(zelena,35)
43  nasobnaCervena = pridejMultiplierKCislum(cervena,2)
44  nasobnaCerna = pridejMultiplierKCislum(cerna,2)
```

Obrázek 3 Definování barev čísel a přidávání násobků

Další část je moje implementace metody Monte Carlo, která vygeneruje podle počtu iterací číslo mezi 0–36. Podle vylosovaného čísla zjistí, jestli se číslo nachází v seznamu výherních čísel a pokud ano tak zjistí jaký násobič má číslo mít. Další krok je vložení pokusu do seznamu postupList kde každý prvek je tuple obsahující, jestli pokus vyhrál a s jakým násobičem. Následuje uložení čísel pro úspěšné a neúspěšné pokusy, přičemž se rozřadí buď do slovníku shots (úspěšné pokusy) nebo shotsMissed (neúspěšné pokusy). Ve slovnících jdou vidět specificky vylosovaná čísla a kolikrát byly vylosována. Tyto slovníky se dají použít pro další analýzu, ale prozatím je jen vypisuji v konzoli.

```
PS E:\Github\SeminarniPraceKMSW>  & 'c:\Users\verne\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\verne\.vscode-insiders\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '64762' '--' 'e:\Github\SeminarniPraceKMSW\7 - Metoda Monte Carlo\ruletaMonteCarlo.py'
{0: 286}
{15: 281, 14: 265, 19: 269, 36: 300, 11: 275, 22: 312, 9: 281, 4: 263, 30: 268, 32: 265, 26: 272, 6: 275, 16: 268, 18: 253, 10: 284, 31: 278, 3: 257, 17: 285, 34: 281, 27: 243, 1: 285, 5: 288, 24: 241, 29: 240, 33: 250, 2: 253, 12: 263, 7: 273, 23: 281, 21: 277, 25: 276, 28: 254, 20: 267, 35: 240, 13: 279, 8: 272}
Výherní šance je: 2.86%
```

Obrázek 4 Slovníky s pokusy vypsány v konzoli (zelená jsou úspěšné, červené jsou neúspěšné)

```
47  def monte_Carlo(pocet_iteraci, vyherniCisla :list):
48      postupList = list()
49      shots = dict()
50      shotsMissed = dict()
51      for i in range(pocet_iteraci):
52          shot = randint(0,36)
53
54          if shot in map(lambda x: x.vyherniCislo, vyherniCisla ):
55              postupList.append((True, next((vyherniCislo for vyherniCislo in vyherniCisla if vyherniCislo.vyherniCislo == shot),2).multiplier))
56              if shot in shots:
57                  shots[shot] += 1
58              else:
59                  shots.update({shot:1})
60          else:
61              postupList.append((False, None))
62              if shot in shotsMissed:
63                  shotsMissed[shot] += 1
64              else:
65                  shotsMissed.update({shot:1})
66      print(colorama.Fore.GREEN + str(shots))
67      print(colorama.Fore.RED + str(shotsMissed))
68      return postupList
```

Obrázek 5 Metoda Monte Carlo

Poslední část je funkce na vyhodnocení strategie. Ta se skládá z dvou částí. Z přípravy grafu a vytvoření grafu. V přípravě grafu zavolá metodu Monte Carlo s počtem pokusů a výsledky metody (trefy a násobky) vloží do objektů typu cisloGrafu na zpracování do grafu.

```
95  def vyhodnotStrategii(vyherni_cisla,pokusu = 10000):
96      sazba = 50
97      postupList = monte_Carlo(pokusu, vyherni_cisla)
98      graf = list()
99      vydelanePenize = 0
100     i = 0
101     pocetHitu = 0
102
103     for (trefil,multiplier) in postupList:
104         if trefil:
105             vydelanePenize += multiplier * sazba
106             pocetHitu+=1
107
108         vydelanePenize -= sazba
109
110         graf.append(cisloGrafu(i,vydelanePenize,pocetHitu,sazba))
111         i += 1
```

Obrázek 6 Začátek funkce na vyhodnocení strategie

Zde se vytváří graf. Jde jen o vytáhnutí specifických dat ze seznamu a vložení do matplotlib knihovny s nějakým nastavením vzhledu. Na konci jen vypisuji do konzole procentuální úspěšnost sázek.

```
    fig, ax = plt.subplots()


    textVyhernichCisel = "Výherní čísla: "+",".join(str(vyherni_cislo.vyherniCislo)for vyherni_cislo in vyherni_cisla)

    ax.plot( list(map(lambda x: x.i,graf)), list(map(lambda x: x.vydelanePenize,graf)), linewidth=2.0,color="Green")
    ax.set_title(f"{textVyhernichCisel}\nVydělané peníze při strategii sázení na čísla se sazbou 50$: " + str(graf[len(graf)-1].vydelanePenize))
    ax.set_xlabel("Počet pokusů")
    ax.set_ylabel("Profit")
    ax.yaxis.set_major_formatter(lambda x, y: "{:n}".format(x)+"$")
    print("Výherní šance je: " +str(pocetHitu / i * 100) + "%")
    plt.show()
```

Obrázek 7 Část vyhodnocovací funkce na vytvoření grafu

Na konci jsem jen zavolal funkce vyhodnotStrategii, aby simulace vyhodnotila mé strategie při 10000 pokusech a 100 pokusech. Každý výsledný graf bude obsahovat jiný výsledek, kvůli separátním voláním metod Monte Carla.

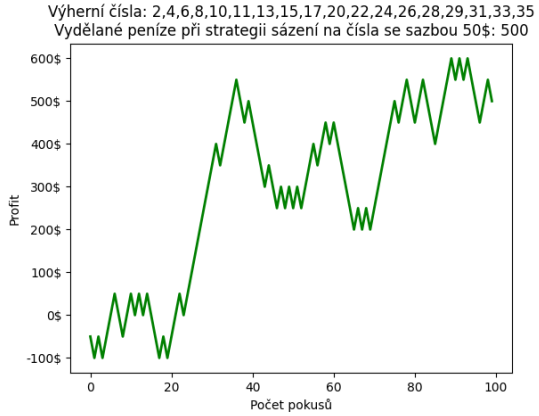
```
vyhodnotStrategii(nasobnaZelena)
vyhodnotStrategii(nasobnaCervena)
vyhodnotStrategii(nasobnaCerna)

vyhodnotStrategii(nasobnaZelena,100)
vyhodnotStrategii(nasobnaCervena,100)
vyhodnotStrategii(nasobnaCerna,100)
```

Obrázek 8 Zavolání vyhodnocení strategií při 10000 a 100 pokusech

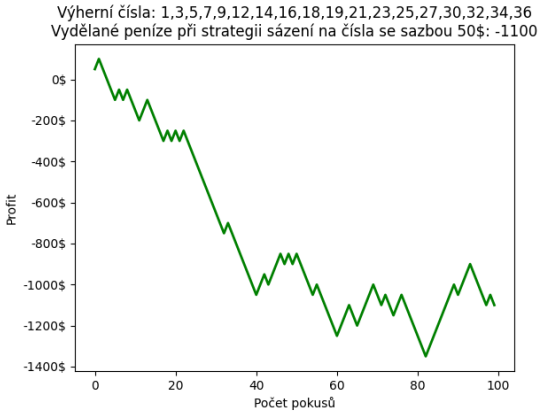
Výsledné grafy

Grafy pro 100 pokusů:

Lze vidět, že strategie sázení na černé čísla se dostala do profitu. To samé se nedá říct o sázkách na červené čísla a na nulu. 100 pokusů je ovšem velice málo a odchylky mezi výsledky simulací mohou být veliké. Proto mám druhý balíček grafů pro 10000 pokusů.



Obrázek 9 Červené sázky pro 100 pokusů

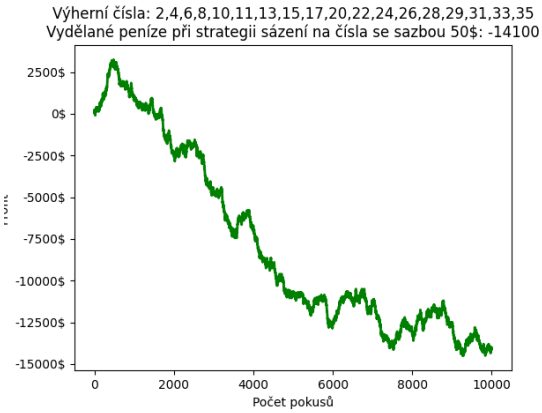


Obrázek 10 Černé sázky pro 100 pokusů

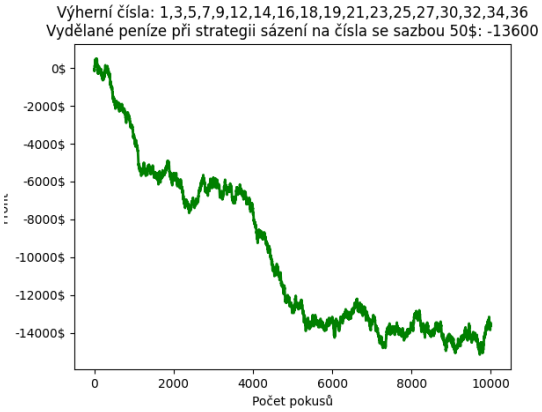
Obrázek 11 Zelené sázky pro 100 pokusů

Grafy pro 10000 pokusů:

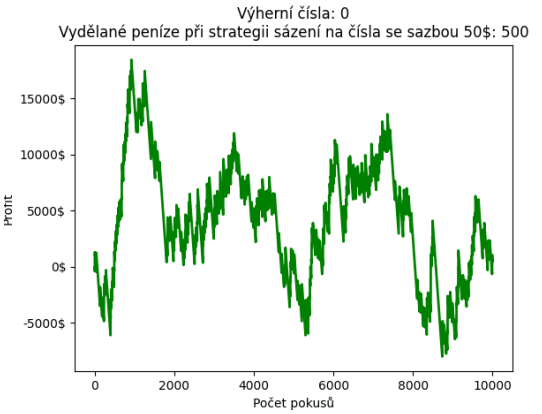
Ve výsledných grafech lze vidět, že v dlouhodobým hledisku sázení na červené a černé je ztrátové. Tato ztráta bude spočívat v tom že pokud padne nula, tak nevyhrává ani jedna z těchto strategií. Strategie sázení na nulu je na pokraji toho že se vyplatí, ale kvůli nízké pravděpodobnosti výhry ale obrovské množství vyhraných žetonů při výhře lze vidět obrovský rozptyl v profitu v průběhu simulace.



Obrázek 12 Červené sázky pro 10000 pokusů



Obrázek 13 Černé sázky pro 10000 pokusů



Obrázek 14 Zelené sázky pro 10000 pokusů

Závěr

Na závěr lze z těchto grafu vyčíst, že sázení na ruletu se nevyplácí (minimálně u uvedených strategií), ale to platí u většiny hazardních her.