Umelá inteligencia
Zadanie 3B
Hľadanie pokladu

Meno: Tomáš Rafaj
Cvičenie: Štvrtok 14:00

Algoritmus

1. Krok - Inicializácia jedincov

Vygenerovanie náhodných jedincov. Vytvorí sa zadané množstvo objektov typu jedinec, kde ich init je vlastne štartovný fitness 1000 a ich bunky sa vygenerujú od 0 do 255, čiže 0 je binary 0000 0000 a 255 je binary 1111 1111

```
class Jedinec(object):
    def __init__(self):
        self.vFitness = 1000
        self.pamat = []
        for i in range(64):
            self.pamat.append(random.randint(0, 255))
```

2. Krok - Vygenerovanie Fitness pre každého Jedinca z kroku 1

Vyznačil som daný krok červeným rámom, kde vidno, že v prípade, že sa jedná o prvú generáciu resp generáciu 0, ktorú sme vytvorili tak vypočítame jej fitness pre každého a v prípade, že sa nám podarilo nájsť prvého jedinca, ktorý nájde všetky poklady (ak funkcia vráti 2). Vypíšeme na ktorú generáciu sa nám to podarilo urobiť a následne pošleme do stroju (funkcia search_solution) jedinca, ktorý toto dokázal a vypíšeme jeho cestu. Ukončíme hľadanie a v GUI sa opýtame užívateľa, čo chce robiť ďalej.

```
for pocetGen in range(pop_count):
    if pocetGen == 0:
        for i in range(POCET_JEDINCOV):
            if search_solution(my_population[i], False) == 2:
                print('Pocet generacii:')
                print(pocetGen + 1)
                print('Cesta')
                search_solution(my_population[i], True)
                notfound = False
                break
    else:
        if notfound is True:
            my_population = krizenie(my_population, int(selection), elitarism)
            for i in range(POCET_JEDINCOV):
                if search_solution(my_population[i], False) == 2:
                    print('Pocet generacii:')
                    print(pocetGen + 1)
                    print('Cesta')
                    search_solution(my_population[i], True)
                    notfound = False
                    break
```

3. Krok - Generovanie nových populácií, kríženie a mutovanie

Na obrázku vyššie môžeme vidieť v prípade, že z prvej / nultej generácie nebude nájdený jedinec, ktorý našiel všetky poklady tak urobíme krizenie na základe argumentov, ktoré si zvolí užívateľ t.j., či selekciu pomocou rulety alebo turnaja a či má program posunúť cca 10% elitných jedincov. Zvolil som 10% z dôvodu, že minimum jedincov je 20, čo sú 2 jedincovia z 20. Po vytvorení novej populácie opätovne posielame jedincov postupne do stroja, kde sa snažíme nájsť všetky poklady, robím až dovtedy pokiaľ nedosiahneme max počet generácií alebo neminieme všetkých jedincov. V prípade úspechu vypíšeme cestu jedinca a jeho číslo generácie. V prípade neúspechu pokračujeme na krok 4

4. Krok

V prípade, že sa nám nepodarilo nájsť jedinca, ktorý by našiel všetky príklady tak stroj počas behov ukladá aktuálne najlepší fitness jeho pôvodný gén, následne teda vypíšem fitness doposiaľ najlepšieho nájdeného jedinca a jeho pôvodnú konfiguráciu 64 buniek. Následne vypíše jeho cestu.

```
if notfound is True:
    print('Nenajdene riesenie so vsetkymi. Ukoncujem hladanie.')
    print('Najlepsi najdedny fitness:')
    print(CURRENT_BEST_FITNESS)
    print('Najlepsi najdedny jedinec:')
    print(str(CURRENT_BEST_JEDINEC)+'\n')
    temp_jedinec = Jedinec()
    temp_jedinec.set_data(CURRENT_BEST_JEDINEC)
    search_solution(temp_jedinec, True)
```

5. Krok

Ukončenie a čakanie na vstup používateľa.

```
input_continue = input('Mam pokracovat ? Y - Ano, N - Nie\n')
if input_continue == 'Y':
    CURRENT_BEST_FITNESS = 0
    CURRENT_BEST_JEDINEC = []
    main_function()
elif input_continue == 'N':
    return 0
```

Virtuálny stroj - funkcia search_solution

Inicializácia

Funkcia si na začiatku definuje globálne premenné, kde odkladá najlepšiu konfiguráciu a najlepší fitness. Taktiež si uloží nie pozmenenú konfiguráciu jedinca, pretože to je samo modifikujúci program, kde hľadá a vypočítava fitness daného jedinca.
Potom si urobíme inicializácie typov inštrukcií :

- Inkrementácia 00xxxxxx, čiže po AND 192(11000000) ostane 0
- Dekrementácia 01xxxxxx, čiže po AND 192(11000000) ostane 64
- Jump 10xxxxxx, čiže po AND 192(11000000) ostane 128
- Print 11xxxxxx, čiže po AND 192(11000000) ostane 192

Definujeme štartovnú pozíciu X = 6, Y = 3

Logika a vykonanie

Následne začne vykonávať inštrukcie až pokým ich nie je 500.
V prípade inkrementácie, zvyšuje obsah bunky objektu.
V prípade dekrementácie, znižuje obsah bunky objektu.

Sú ošetrené aj prípady, že pripočítame k 255 (11111111) + 1, to sa zachová tak, že vznikne 00000000 a naopak. V prípade posunov za index, t.j. budeme sa chcieť posunúť doprava ale už sme na poslednom indexe sa vrátime na index 0. V prípade jump-u iba meníme aktuálny index, pričom je tiež ošetrené aby sme neskočili preč. V prípade výpisu mi chvíľku trvalo nájsť najsprávnejšie riešenie orátania 1 a nakoniec som si nechal poradiť zadaním a vyriešil som to modulo 4 (ktoré generuje 4 možnosti {0, 1, 2, 3}. Pričom vlastne rieši posledné 2 bity v čísle, čísla deliteľné 4 majú na konci 00, čísla mod 1 majú 01, čísla mod 2 majú 10 ale čísla mod 3 majú 11.

Rátanie Fitness

Za každý krok je mínus 1 a za každý nájdený poklad je plus 1000, pričom začíname na Fitness hodnote 1000.

Ošetrenie vybočenia

V prípade, že vybehneme von z mriežky, končíme hľadanie strojom a pokračujeme na ďalšie.

Prípad nájdenia všetkých pokladov

Ukončíme generovanie a hľadanie všetkého, vypíšeme postupnosť.

Maximálny počet inštrukcií

Maximum je ošetrené počtom max 500 inštrukcií.

Tvorba nových generácií a kríženie

- Nadvýber
  - S elitarizmom
    - 10% najlepších jedincov sa vloží do novej generácie
  - Bez elitarizmu
- 2 typy selekcie
  - Ruleta
    - Každému pridelíme toľko políčok v ruletu, koľko zodpovedá jeho fitness-u, následný náhodný výber z rulety
  - Turnaj
    - 3 jedinci sú v turnaji, zoberie sa najlepší z nich
- Kríženie (R1 + R2 = Kríženec)

```
for k in range(how_much_parent1 - pocet_elit):
    krizenec.set_bunka(k + pocet_elit, final_parent1.pamat[k + pocet_elit])
for k in range(64 - how_much_parent1 - pocet_elit):
    krizenec.set_bunka(k + how_much_parent1 + pocet_elit,
                       final_parent2.pamat[k + how_much_parent1 + pocet_elit])
```

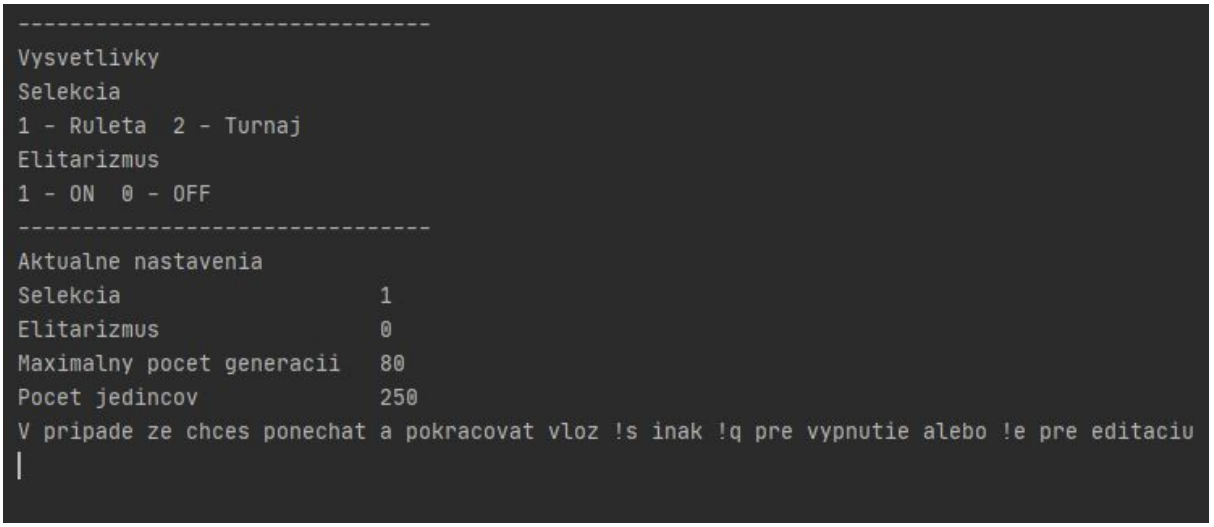
- Mutácia

```
if random.randint(1, 100) < 5:
    index = random.randint(0, 63)
    mutacia = random.randint(0, 255)
    krizenec.set_bunka(index, mutacia)

    index = random.randint(0, 63)
    mutacia = random.randint(0, 255)
    krizenec.set_bunka(index, mutacia)

    index = random.randint(0, 63)
    mutacia = random.randint(0, 255)
    krizenec.set_bunka(index, mutacia)

    index = random.randint(0, 63)
    mutacia = random.randint(0, 255)
    krizenec.set_bunka(index, mutacia)

    index = random.randint(0, 63)
    mutacia = random.randint(0, 255)
    krizenec.set_bunka(index, mutacia)
```

Užívateľ si môžeme na základe GUI vybrať, či chce vykonávať s / bez elitarizmu. Taktiež si vie vybrať spôsob výberu z rulety a turnaju. Kríženie prebieha pomocou skríženia rodiča 1 a rodiča 2, pričom časť rodiča 1 a rodiča 2 nie je rozdelená rovnomerne ale náhodne, generuje sa akú časť bude mať rodič 1 a rodič 2. Kríženie je nastavené na šancu 5 zo 100, čiže 5%, pričom vznikne úplná zmena genetickej informácie v max až 5 bunkách.

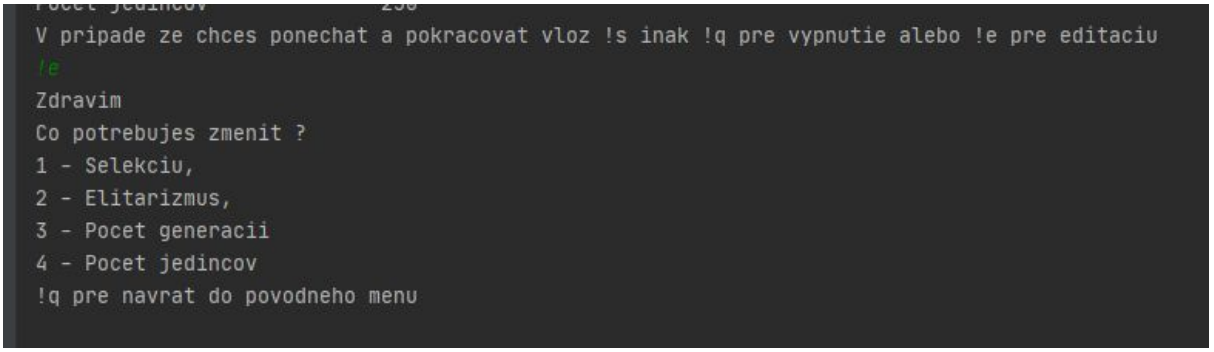
GUI - užívateľské rozhranie cez konzolu

Pri spustení programu vypíše :

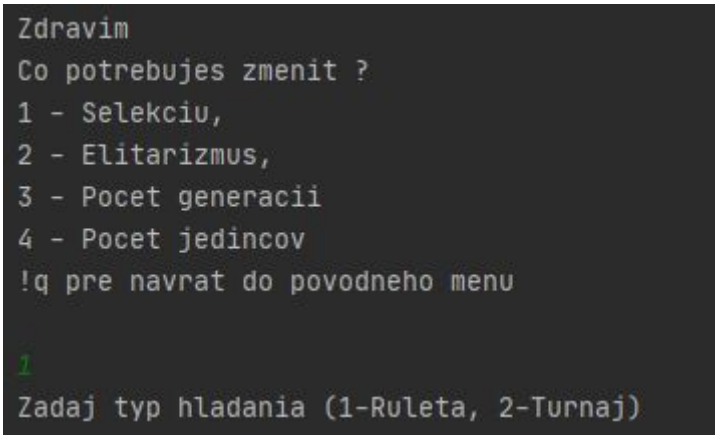


Kde môžeme zadať :
!s pre štart vyhľadávania s parametrami vyššie
!q pre vypnutie programu
!e pre editáciu programu

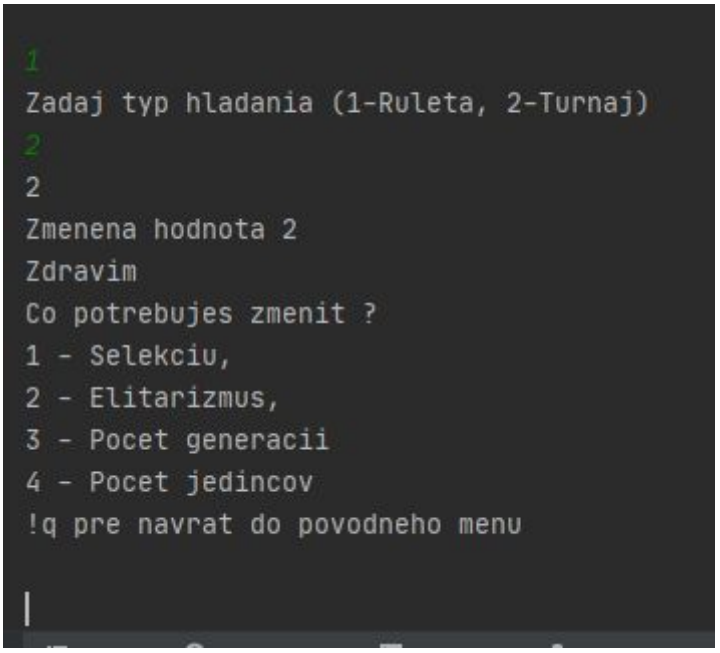
Po stlačení !e



Príde otázka, čo potrebujem zmeniť, zadaním čísla a stlačením enteru sa otvorí nové menu, napr.



Po vložení vhodného parametru sa to zmení inak vypíše chybu a končí program, v prípade, že chcem ísť do spúšťacieho menu napíšeme !q



```
Zdravim
Co potrebujes zmenit ?
1 - Selekciu,
2 - Elitarizmus,
3 - Pocet generacii
4 - Pocet jedincov
!q pre navrat do povodneho menu

!q
--------------------------------
Vysvetlivky
Selekcia
1 - Ruleta  2 - Turnaj
Elitarizmus
1 - ON  0 - OFF
--------------------------------
Aktualne nastavenia
Selekcia                    2
Elitarizmus                 0
Maximalny pocet generacii   80
Pocet jedincov              250
V pripade ze chces ponechat a pokracovat vloz !s inak !q pre vypnutie alebo !e pre editaciu
```

Takto už vyzerá navrátené menu.
Po stlačení !s

```
U
D
R
R
L
U
Poklad X 2 Y 5
D
R
R
L
U
D
R
D
Najdene vsetky poklady. Koncim program. X 4 Y 7
Mam pokracovat ? Y - Ano, N - Nie
```

Programový výstup vyzerá nasledovne, kde po stlačení ‘Y’ sa vrátime do štartovného menu a pri stlačení ‘N’, ukončíme program.

Zhodnotenie vlastností

Testovanie som sa rozhodol urobiť tak, že si spravím úspešnosť z 2x 10 náhodných zbehnutí pre 4 typy.

1. Ruleta bez elitarizmu
2. Ruleta s elitarizmom
3. Turnaj bez elitarizmu
4. Turnaj s elitarizmom

Testovanie A (veľa jedincov, málo generácií)

Konfigurácia	
Maximalny pocet generacii	80
Pocet jedincov	250

Ruleta bez elitarizmu

Zbehnutie 1	našlo všetky poklady - generácia 28
Zbehnutie 2	našlo všetky poklady - generácia 20
Zbehnutie 3	našlo všetky poklady - generácia 38
Zbehnutie 4	našlo všetky poklady - generácia 22
Zbehnutie 5	nenašlo, najlepší fitness 4975
Zbehnutie 6	nenašlo, najlepší fitness 4974
Zbehnutie 7	našlo všetky poklady - generácia 16
Zbehnutie 8	nenašlo, najlepší fitness 4984
Zbehnutie 9	nenašlo, najlepší fitness 4981
Zbehnutie 10	našlo všetky poklady - generácia 58

Ruleta s elitarizmom

Zbehnutie 1	nenašlo, najlepší fitness 4970
Zbehnutie 2	našlo všetky poklady - generácia 27
Zbehnutie 3	nenašlo, najlepší fitness 4972
Zbehnutie 4	nenašlo, najlepší fitness 4975
Zbehnutie 5	našlo všetky poklady - generácia 8
Zbehnutie 6	nenašlo, najlepší fitness 4959
Zbehnutie 7	našlo všetky poklady - generácia 28
Zbehnutie 8	našlo všetky poklady - generácia 70
Zbehnutie 9	nenašlo, najlepší fitness 4983
Zbehnutie 10	našlo všetky poklady - generácia 12

Turnaj s elitarizmom

Zbehnutie 1	nenašlo, najlepší fitness 4966
Zbehnutie 2	našlo všetky poklady - generácia 77
Zbehnutie 3	našlo všetky poklady - generácia 7
Zbehnutie 4	našlo všetky poklady - generácia 35
Zbehnutie 5	nenašlo, najlepší fitness 4979
Zbehnutie 6	nenašlo, najlepší fitness 4968
Zbehnutie 7	nenašlo, najlepší fitness 4970
Zbehnutie 8	našlo všetky poklady - generácia 16
Zbehnutie 9	našlo všetky poklady - generácia 16
Zbehnutie 10	nenašlo, najlepší fitness 4973

Turnaj bez elitarizmu

Zbehnutie 1	nenašlo, najlepší fitness 4976
Zbehnutie 2	nenašlo, najlepší fitness 4916
Zbehnutie 3	našlo všetky poklady - generácia 9
Zbehnutie 4	našlo všetky poklady - generácia 5
Zbehnutie 5	nenašlo, najlepší fitness 4980
Zbehnutie 6	našlo všetky poklady - generácia 10
Zbehnutie 7	nenašlo, najlepší fitness 4983
Zbehnutie 8	našlo všetky poklady - generácia 17
Zbehnutie 9	nenašlo, najlepší fitness 4980
Zbehnutie 10	nenašlo, najlepší fitness 4968

Testovanie B (málo jedincov, veľa generácií)

Konfigurácia

Maximalny pocet generacii	600
Pocet jedincov	20

Ruleta bez elitarizmu

Zbehnutie 1	nenašlo, najlepší fitness 4967
Zbehnutie 2	nenašlo, najlepší fitness 4968
Zbehnutie 3	nenašlo, najlepší fitness 3982
Zbehnutie 4	našlo všetky poklady - generácia 150
Zbehnutie 5	nenašlo, najlepší fitness 4974
Zbehnutie 6	nenašlo, najlepší fitness 4985
Zbehnutie 7	nenašlo, najlepší fitness 4957
Zbehnutie 8	nenašlo, najlepší fitness 3987
Zbehnutie 9	nenašlo, najlepší fitness 2993
Zbehnutie 10	nenašlo, najlepší fitness 4967

Ruleta s elitarizmom

Zbehnutie 1	nenašlo, najlepší fitness 4975
Zbehnutie 2	našlo všetky poklady - generácia 208
Zbehnutie 3	našlo všetky poklady - generácia 187
Zbehnutie 4	nenašlo, najlepší fitness 4981
Zbehnutie 5	našlo všetky poklady - generácia 293
Zbehnutie 6	nenašlo, najlepší fitness 4978
Zbehnutie 7	našlo všetky poklady - generácia 250
Zbehnutie 8	našlo všetky poklady - generácia 135
Zbehnutie 9	nenašlo, najlepší fitness 4977
Zbehnutie 10	našlo všetky poklady - generácia 66

Turnaj bez elitarizmu

Zbehnutie 1	nenašlo, najlepší fitness 4974
Zbehnutie 2	nenašlo, najlepší fitness 4968
Zbehnutie 3	nenašlo, najlepší fitness 4982
Zbehnutie 4	nenašlo, najlepší fitness 2993
Zbehnutie 5	nenašlo, najlepší fitness 3983
Zbehnutie 6	nenašlo, najlepší fitness 3979
Zbehnutie 7	našlo všetky poklady - generácia 23
Zbehnutie 8	nenašlo, najlepší fitness 4979
Zbehnutie 9	nenašlo, najlepší fitness 4948
Zbehnutie 10	našlo všetky poklady - generácia 15

Turnaj s elitarizmom

Zbehnutie 1	našlo všetky poklady - generácia 382
Zbehnutie 2	našlo všetky poklady - generácia 149
Zbehnutie 3	našlo všetky poklady - generácia 263
Zbehnutie 4	našlo všetky poklady - generácia 186
Zbehnutie 5	našlo všetky poklady - generácia 261
Zbehnutie 6	nenašlo, najlepší fitness 4983
Zbehnutie 7	našlo všetky poklady - generácia 221
Zbehnutie 8	našlo všetky poklady - generácia 379
Zbehnutie 9	nenašlo, najlepší fitness 4969
Zbehnutie 10	našlo všetky poklady - generácia 55

Zhrnutie testov

Po zbehnutí a testovaní 2 očividne rozličných scenárov môžeme jasne vidieť, že elitarizmus extrémne zvýši účinnosť pri väčšom počte generácií hlavne u turnaju. kde dosiahol až 80% úspešnosť a ostatné najlepšia hľadania boli 4 poklady z 5. Zlepšenie z 10% na 50% úspešnosť môžeme vidieť aj pri rulete, kde väčší počet generácií jasné nahráva k prejaveniu elitarizmu. Pri malom počte generácii sme ale mohli vidieť, že výsledky elitarizmu mali zanedbateľné zlepšenie, pretože aj bez boli už veľmi slušné nájdenia všetkých pokladov s veľkým počtom jedincov. Avšak je tam vidieť zlepšenie napríklad v znížení rýchlejšieho nájdenia správneho riešenia, kde vidno, že elity spôsobili skoršie nájdenie v generácii (t.j. skoršia generácia populácie)

Čo by sa dalo ešte vylepšiť na zadaní ?

Dali by sa rozhodne urobiť viaceré druhy selekcie okrem rulety a turnaju, ktoré by mohli do väčšej hĺbky ukázať použitie a funkčnosť elitarizmu.

Dal by sa navrhnúť komplikovanejší virtuálny stroj, ktorý by mohol mať väčšiu funkcionalitu, avšak to rozhodne nebolo cieľom zadania, prípadne aj nepriama adresácia.