HRYTSYNA ANASTASIIA-SOLOMIIA (ID: 105323)

Zenová záhrada (zadanie č. 3a)

❖ Riešený problém (zadanie)

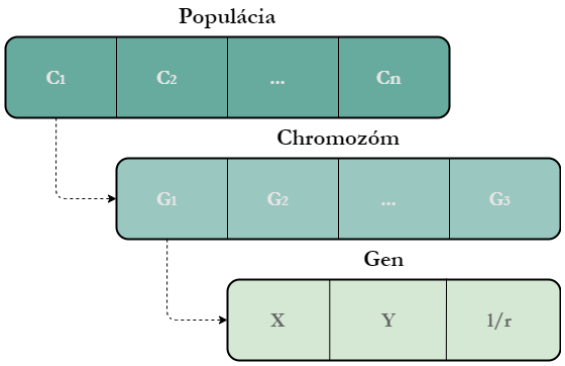
Zenová záhradka je plocha vysypaná hrubším pieskom (drobnými kamienkami). Obsahuje však aj nepohyblivé väčšie objekty, ako napríklad kamene, sochy, konštrukcie, samorasty. Mních má upraviť piesok v záhradke pomocou hrablí tak, že vzniknú súvislé pásy. Pásy môžu ísť len vodorovne alebo zvislo, nikdy nie šikmo. Začína vždy na okraji záhradky a ťahá rovný pás až po druhý okraj alebo po prekážku. Na okraji – mimo záhradky môže chodiť ako chce. Ak však príde k prekážke – kameňu alebo už pohrabanému piesku – musí sa otočiť, ak má kam. Ak má voľné smery vľavo aj vpravo, je jeho vec, kam sa otočí. Ak má voľný len jeden smer, otočí sa tam. Ak sa nemá kam otočiť, je koniec hry. Úspešná hra je taká, v ktorej mních dokáže za daných pravidiel pohrabať celú záhradu, prípade maximálny možný počet políčok. Výstupom je pokrytie danej záhrady prechodmi mnícha.

❖ Opis riešenia

- **Implementáčne prostredie**

Moje riešenie som implementovala pomocou programovacieho jazyku Python vo vývojovom prostredí PyCharm 2020.2.3. Naviac, použila som knižnice random a time (pre výpočty) a matplotlib (pre kreslenie zobrazení). Myslím si, že práve pomocou objektovo orientovaného programovania dá sa efektívnejšie a rýchlejšie vyriešiť danú úlohu.

- **Gény, chromozómy, populácie**



Najmenší prvok môjho genetického algoritmu je **gen**. On reprezentuje polohu mnícha v záhrade (jeho štartovaciu pozíciu na mape). Teda každý gén má svoje štartovacie súradnice (X, Y) a preferovaný smer (doprava "r" alebo doľava "l") otočenia/pohybu. V svojom programe ja generujem gény na celom obvode (na všetkých stranách), potom ich poradie miešam a vyberám max povolený počet génov v chromozóme.

Skupina rôznych génov (ktorý sa nezopakujú) formuje celý **chromozóm**, a niekoľko chromozómov (toľko, koľko je daný population_size) vytvárajú **populáciu**. Každý chromozóm sa snaží nájsť úspešnú cestu mnícha cez záhradku. Je jedna mapa, na ktorej sa zapisujú pohyby mnícha (čiže spúšťajú sa rôzne gény). Ak gén nenašiel žiadnu cestu (a narazil sa na prekážku), tak zabudnem na tu cestu. Keď už sa nezostanú žiadne gény, tak vrátime fitness pre chromozóm. Ak on sa nerovná max fitnessu (cesta nebola úplne nájdená), tak vynulujeme si mapu a pokračujeme s ďalším chromozómom. Inak záhradku vyprintujeme.

- **Mních, mapa**

Mních a jeho pohyb sú reprezentované génmi chromozómu. V každom géne sa randomne zvolí smer, kam je preferovaný pohyb. V randomnom poradí gény sú umiestnené v chromozóme, takže aj postupnosť ich spúšťaní je náhodná. Samotná **mapa** je reprezentovaná 2D array-om. Kamene sú označené ako '-1', voľné políčka ako '0', a iné očíslované poľa sú cesty mnícha. Mapa záhradky sa vytvára iba 1 krát pre lepšiu efektivitu a resetuje sa za každým "neúspešným chromozómom".

- **Fitness**

Každý chromozóm ma svoj pridelený fitness. To hodnotenie rátame na základe množstva úspešne nájdených mníchom ciest (kamene sa nerátajú). Čiže fitness v chromozóme je taký, koľko je kladných políčok v mape (ohodnotených ciest, na ktorých mních vstúpil a vystúpil zo záhradky). Čím viac správnych ciest našiel mních, tým lepší fitness on dostal.

- **Tvorba novej generácie (výber jedincov, crossover, mutácie)**

Na začiatku programu sa vytvorí **prvá generácia** (funkcia createInitialPop( )). Ak riešenie zadania nebolo nájdené, tak pokračujeme vytvárať novú generáciu. Potom budeme zopakovať ten postup až pokiaľ nenájdeme riešenie či neprekročíme max limit počtu generácií.

Čiže program funguje na základe genetického algoritmu, pre správne fungovanie ktorého potrebujeme v 1. kroku si **zvoliť jedinca** (rodiča), ktorý budeme nasledovať. Jeho v mojom programe je možné vybrať jedným z dvoch spôsobov:

1. **Tournament selekcia**

V mojom programe to znamená, že náhodné z populácie si zvolím 3 chromozómy, porovnám ich a rodičom označím ten, ktorý má najvyšší fitness.

2. **Roulette selekcia**

Predstavím si, že fitness chromozómu je zároveň aj pravdepodobnosť, že prvok si zvolíme ako rodič. Teda čím väčší je fitness, tým ten prvok ma väčšiu možnosť stať rodičom. V programe to robím tak, že generujem nejaké randomne číslo od 0 až po sumu fitnesov generácie a vyberám si ten chromozóm, ktorý v sume ma odsek, kde je generované naše randomne číslo.

Tým pádom zopakujem si nejakú selekciu 2 krát a vyberiem dvoch rodičov. Z dvoch rodičov ďalej vytvorím dvoch detí pomocou **kríženia** s pravdepodobnosťou crossover_rate. Jeho aplikujem tak, že každé dieťa poskladám z génov jeho rodičov (náhodné vybraných či z 1. alebo 2.). Tie chromozómy, ktoré sa nekrížia, budú iba náhodné prekopírované a vložené do novej populácie.

Ďalším krokom bude zmutovať našich vytvorených deti. **Mutáciu** robím tromi rôznymi spôsobmi s pravdepodobnosťou ~0,33% každá. **Prvý typ** mutácie - výmena iba smeru génu (teda "r"/"l" s pravdepodobnosťou 50%). **Druhý** - výmena poradia dvoch génov v chromozóme (ich swapovanie). A posledný, **tretí** - výmena génu na úplne nový náhodný generovaný prvok. Naviac, každý gén má pravdepodobnosť mutovať takú, aký je mutation_rate. Postupne s vývojom programu mutation_rate sa mení (napríklad v tom prípade, ak priemerný fitness už v niekoľkých generáciách je rovnaký). Takže celkovo on môže byť v rozsahu od mutation_min až po mutation_max. Keď už pravdepodobnosť mutácie dosiahne maximálnu možnú hodnotu, tak sa to opäť resetne na minimálnu.

Koncovým produktom našich zmien je populácia, ktorá sa skladá z chromozómov. V každom chromozóme(dieťati) sú rodičovské gény s crossoverom a nejakou mutáciou.

- **Používateľské rozhranie**

Po spustení môžeme si zvoliť niekoľko možností vývoja programu (napríklad testovať program alebo spustiť mnícha do záhrady). Oni sa vypisujú v konzole tak, ako je to uvedené na obrázku (jedna z možností). Počas behu programu v konzolu sa taktiež vypisuje jej vývoj, a po vykonaní dostaneme nie len konzolový výpis, a aj graf ciest či porovnaní parametrov (iba vo veľkom teste sa nám zapisujú údaje do .txt). Príklad je znázornení na obrázku.

```
Stlacte   1  , ak chcete otestovat geneticky algoritmus, alebo   2  , ak chcete spustit mnicha do zahradky.
2

Stlacte
1  - ak chcete vyuzit standartnu mapu
2  - ak chcete vytvorit svoju mapu
2

Stlacte   1  , ak chcete vymenit globalne parametre, alebo ine cislo, ak ich chcete nechat defaultne
4

Zadajte sirku zahrady (OX)
12

Zadajte dlzku zahrady (OY)
10

Zadajte pocet kamenov
4

Zadajte ich X Y suradnice
1 2
5 1
4 3
2 4
8 6
9 6
```

Príklad input-u

	0	1	2	3	4	5	6	7	8	9	10	11
0	6	6	5	11	11	4	4	1	12	12	3	13
1	6	6	5	11	11	K	4	1	12	12	3	13
2	10	K	5	5	5	5	4	1	12	12	3	13
3	10	5	5	5	K	5	4	1	12	12	3	13
4	5	5	K	5	5	5	4	1	12	12	3	13
5	4	4	4	4	4	4	4	1	12	12	3	13
6	1	1	1	1	1	1	1	1	K	K	3	3
7	7	7	7	7	7	7	2	2	2	2	2	2
8	8	8	8	8	8	7	2	9	9	9	9	9
9	8	8	8	8	8	7	2	9	9	9	9	9

Príklad output-u

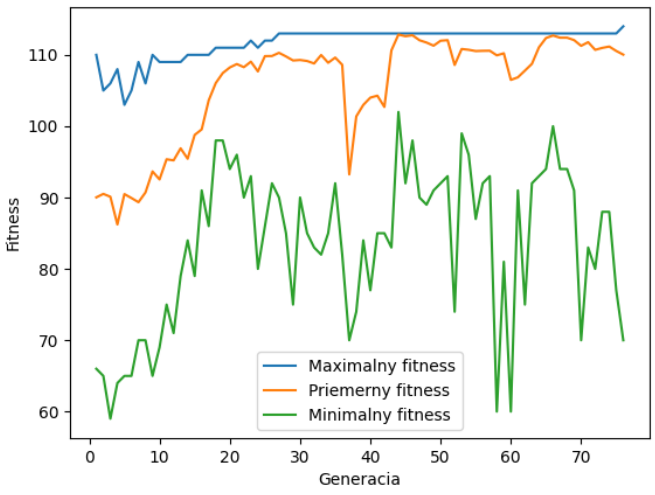
❖ Otestovanie programu

▪ Možnosti nastavenia parametrov

V programe sa dá nastaviť aj globálne parametre, nie iba informácie o záhradke. To sa robí vo funkcie change_parameters( ), ktorá sa vyvoláva za vašim želaním na začiatku programu (pred spustením mnícha po záhrade). **Parametre** sú také:
max_generations - najväčší počet generácií (limit, pre ktoré hľadáme riešenie); randomne kladne číslo
population_size - počet chromozómov v jednej populácií; randomne kladne číslo
crossover_rate - pravdepodobnosť chromozómu byť krížením od rodiča, nie len náhodné vybraným; číslo je v intervale (0, 1)
mutation_min a **mutation_max** - veľkosť možnej meniacej sa mutácie, ak nastaviť druhy parameter na 0, tak mutácia bude konštantná; obidva čísla sú v intervale (0, 1)
parent_selection - typ výberu rodiča; 1 pre Tournament, 2 pre Roulette selekciu

▪ Porovnanie dosahovaných výsledkov pre viacero nastavení parametrov

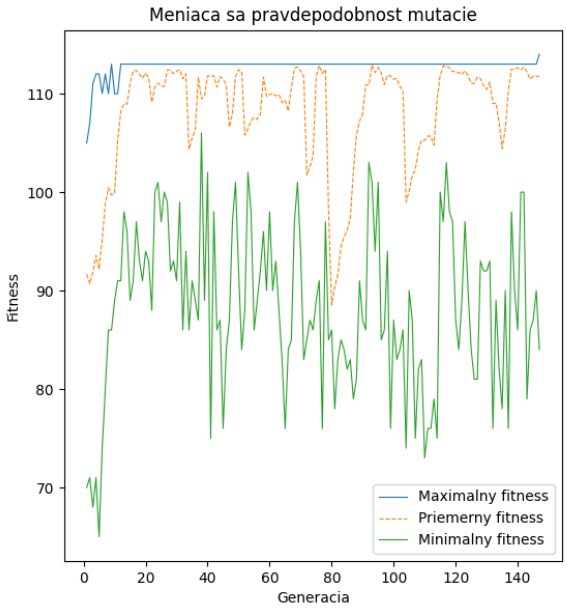
Najprv, svoj kód ja som testovala na **veľkom teste (test #1)**, aby zistiť **aké parametre sú najlepšie** v mojom prípade. Na niekoľko hodín som spustila bežať program a dostala som údaje, ktoré sú uvedené v tabuľke nižšie. Čiže pre každú kombináciu parametrov som skúšala 20 krát zbehnúť program a rátala, koľko úspešných riešení (pre limit 1000) som dostala. Podľa mňa najlepší je výsledok #2 (je označení červeným v tabuľke) a preto tie parametre som pridala ako defaultne globálne v program. Taktiež pre tie parametre som dala zbehnúť aj štandardnej mape a taký graf zmeny fitnessu počas behu programu som dostala: (viď obrázok). Riešenie bolo nájdene cca za 75 generácií.

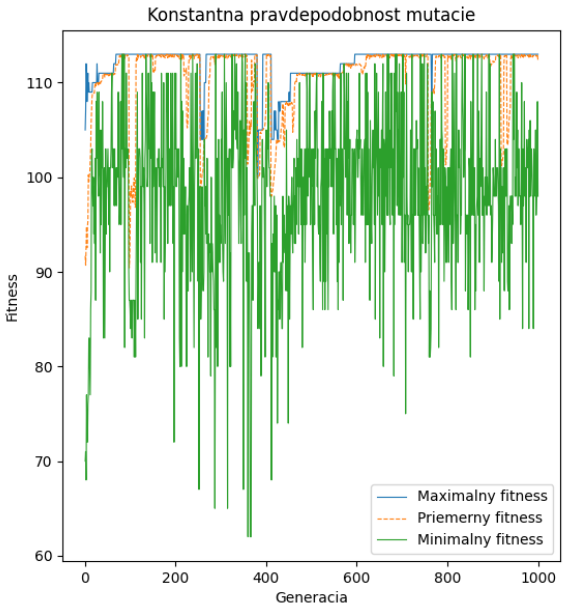


HRYTSYNA ANASTASIIA-SOLOMIIA (ID: 105323)

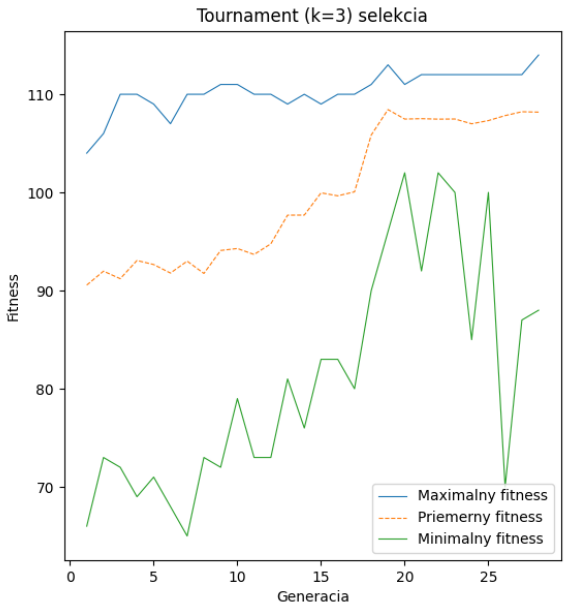
Selekcia rodica	Krizenie	Velkost populacie	Min mutacia	Uspechy	Pocet pokusov	Priemerny pocet generacii
2	0,99	150	0,02	19	20	488
1	0,99	150	0,01	17	20	288,55
1	0,97	150	0,02	17	20	439,45
2	0,97	100	0,01	17	20	516,2
1	0,98	150	0,02	16	20	442,2
1	0,97	100	0,02	16	20	481,95
2	0,99	75	0,01	16	20	506,85
2	0,98	150	0,02	16	20	546,6
1	0,98	100	0,02	15	20	497,95
1	0,98	75	0,02	15	20	505,35
1	0,98	50	0,01	15	20	538,1
2	0,98	150	0,01	15	20	565,8
1	0,97	100	0,01	14	20	385,75
1	0,99	150	0,02	14	20	460,05
1	0,98	75	0,01	14	20	510,15
2	0,99	100	0,01	14	20	636,85
2	0,97	150	0,02	14	20	642,75
2	0,98	100	0,02	14	20	705,75
1	0,98	150	0,01	13	20	481,7
1	0,99	50	0,02	13	20	490,4
1	0,97	150	0,01	13	20	495,2
1	0,97	75	0,01	13	20	519,9
1	0,97	50	0,01	13	20	535,6
2	0,97	150	0,01	13	20	567,35
1	0,98	25	0,01	13	20	581,35
1	0,98	100	0,01	13	20	598,25
1	0,99	100	0,02	13	20	667,35
2	0,97	75	0,02	13	20	728,15
1	0,99	50	0,01	12	20	531,3
1	0,99	100	0,01	12	20	608,1
2	0,97	100	0,02	12	20	630,45
2	0,98	75	0,02	12	20	680,05
2	0,98	75	0,01	12	20	738,6
1	0,98	50	0,02	11	20	599,55
1	0,97	75	0,02	11	20	633,45
2	0,98	50	0,02	11	20	658,9
2	0,99	150	0,01	11	20	662,35
1	0,99	25	0,01	11	20	759,7
1	0,97	50	0,02	10	20	628,1
1	0,99	25	0,02	10	20	658,55
1	0,97	25	0,02	10	20	694,15
1	0,99	75	0,01	10	20	717,65
2	0,97	75	0,01	10	20	760,9
2	0,99	50	0,01	9	20	728
2	0,99	100	0,02	9	20	803,15
2	0,98	100	0,01	9	20	804,55
1	0,97	25	0,01	8	20	756,35
1	0,99	75	0,02	8	20	759
2	0,99	75	0,02	8	20	779
2	0,97	50	0,01	8	20	792,35
2	0,98	50	0,01	7	20	779,5
1	0,98	25	0,02	7	20	792,6
2	0,99	50	0,02	5	20	838,75
2	0,97	50	0,02	5	20	888,75
2	0,97	25	0,01	2	20	934,95
2	0,99	25	0,01	1	20	950,45
2	0,98	25	0,02	1	20	959,35
2	0,99	25	0,02	1	20	968,75
2	0,97	25	0,02	1	20	975,8
2	0,98	25	0,01	1	20	982,75

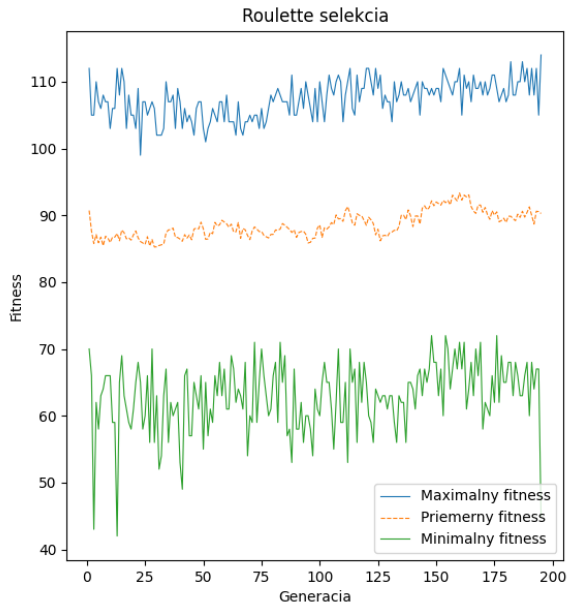
Ako je to vidieť na predbežnom grafe s fitnessom - jeho priemerná hodnota niekedy sa dôrazne meny a "skáče". Je to práve kvôli tomu, že **pravdepodobnosť mutácie** sa zvyšuje a mení sa, takže s časom pribúda viac nových prvkov. Taktiež som skúsila spustiť svoj algoritmus aj za predpokladu konštantnéj mutácie (mapa a seed tiež sú tie isté, takže všetky podmienky sú rovnaké). Vo výsledku som dostala tieto dva grafy (viď obrázok). Z nich vyplýva, že za podmienok meniacej sa mutácie algoritmus nájde svoje riešenie oveľa rýchlejšie, ako v prípade stálej pravdepodobnosti mutácie (kde riešenie vôbec nebolo nájdené v našom prípade), čo moc zlepšuje efektivitu môjho algoritmusu.





Nakoniec chcela som zistiť ako **týp výberu rodiča** ovplyvňuje riešenie. Tak isto ako aj pred tým, spustila svoj program za rovnakých podmienok a dostala som 2 grafy. Je to hneď vidieť, že Tournament selection funguje o dosť lepšie, ako Roulette selection. Vo väčšine prípadov Tournament nájde riešenie skôr. Od časti možno je to kvôli tomu, že pomocou Tournament selekcii je väčšia šanca zahrnúť aj tie prvky, ktoré majú menší fitness (ako obyčajne, nezáleží na fitnesse až tak) a že časovo to vychádza rýchlejšie, lebo nemusím prechádzať celý zoznam všetkých fitnessov (ako to robím v Roulette). Inak to vysvetliť neviem 🙂





❖ Zhodnotenie

V celom môj program funguje korektne a správne. Vo väčšine prípadov on dokaze nájsť riešenie do zadaného limitu. Snažila som sa ho vylepšiť čonajviac, aby on fungoval aj dosť rýchlo (prepisovala som to z tried, pridavala vylepšenia a hľadala najlepšie parametre). Zistila som, že meniaca sa mutácia a Tournament selekcia o dosť zlepšujú beh programu. Našla som aj priemerné dobre globálne parametre, ale osobne si myslím, že ešte by sa zdalo spustiť môj program na väčších počtoch testov (a aj s väčšou veľkosťou) a vtedy by som sa viac priblizovala k najlepšiemu riešeniu.