Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológii

15. November 2021

Umelá inteligencia

Zadanie č. 3a- Zenová záhrada

Evolučný algoritmus

Roland Vdovják
id: 110912

Obsah

Roland Vdovják
ID: 110912

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 pásy ako na nasledujúcom obrázku.

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. Pokrytie zodpovedajúce presne prvému obrázku (priebežný stav) je napríklad takéto:

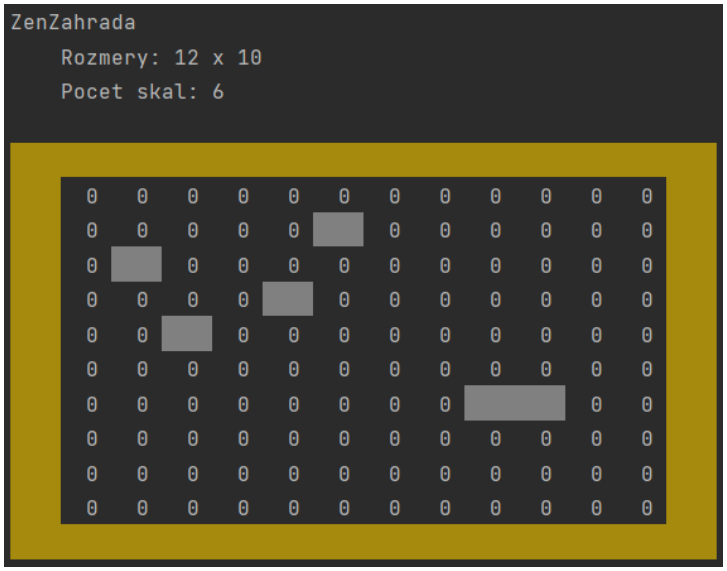
0	0	1	0	0	0	0	0	10	10	8	9
0	0	1	0	0	K	0	0	10	10	8	9
0	K	1	0	0	0	0	0	10	10	8	9
0	0	1	1	K	0	0	0	10	10	8	9
0	0	K	1	0	0	0	0	10	10	8	9
2	2	2	1	0	0	0	0	10	10	8	9
3	3	2	1	0	0	0	0	K	K	8	8
4	3	2	1	0	0	0	0	5	5	5	5
4	3	2	1	0	0	0	11	5	6	6	6
4	3	2	1	0	0	0	11	5	6	7	7

Riešenie

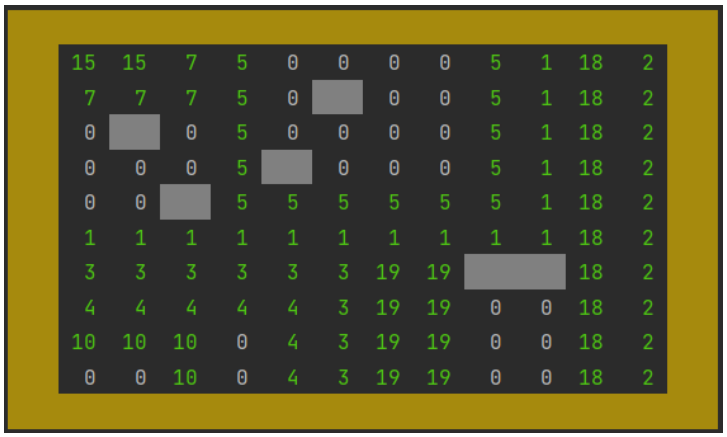
Skôr ako sme sa dostali ku samotnému evolučnému algoritmu sme potrebovali mať naprogramovanú záhradu a mnícha. Mních potom podľa jeho chromozómu pohrabe záhradu a na základe pohrabania dostane ohodnotenie- *fitness*. Jedince sme vyberali pomocou turnaja a elitizmu.

1. Záhrada

Záhradu reprezentujeme pomocou matice o veľkosti x+2 a y+2 (x predstavuje šírku, y predstavuje výšku). Mních vie hrabať záhradu aj o iných rozmeroch ako je v zadaní (12x10). Našim hlavným cieľom je pohrabať záhradu so skalami zo zadania. Aby sme mali prehľad, ako záhrada vyzerá počas jednotlivých generácií, používame funkciu *print_garden()*, ktorá farebne oddelí rôzne prvky v záhrade. Záhrada zo zadania vyzerá v našom prípade takto:



Žltou farbou je reprezentovaný okraj, ktorý slúži ako vstup/výstup mnícha zo záhrady. Sivé bloky vo vnútri záhrady sú kamene. Nepohrabané políčka majú hodnotu „0“, pohrabané menia farbu na zelenú a nadobúdajú hodnotu poradia génu, ktorý ich pohrabal.



2. Mních

Vytvorili sme vlastnú triedu *Monk*:

```
class Monk:
    def __init__(self, chromosome, fitness, garden= None):
        self.chromosome = chromosome
        self.fitness = fitness
        self.garden = garden
```

Obsahuje chromozóm, fitnes a pohrabanú záhradu.

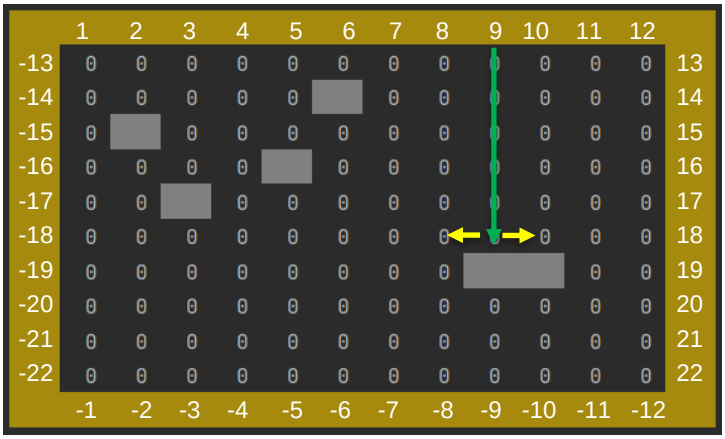
2.1 Chromozóm

Chromozóm je zložený z génov, gény sú reprezentované celým číslom z intervalu

$$< -(x + y),\ x + y >$$

Kde *x* predstavuje šírku a *y* výšku záhrady. Chromozóm má *x+y+1* génov. Každý gén okrem posledného hovorí o tom, ktorý riadok alebo stĺpec záhrady má mních pohrabať. Záporność čísla znamená, že daný stĺpec alebo riadok sa prechádza z opačnej strany. Posledný gén je náhodné číslo, podľa ktorého sa program rozhoduje, ktorým smerom má odbočiť ak narazí na prekážku. Chromozóm plný génov pre záhradu zo zadania môže vyzerať takto:

Úloha chromozómu sa dá vizuálne interpretovať nasledovne. Biele číslo na okraji záhrady označuje riadok stĺpec a smer z ktorého mních začína.



Ak mních zastaví na prekážke (skala alebo pohrabané miesto) musí sa rozhodnúť, ktorým smerom pôjde. Tieto smery môžu byť maximálne dva (žlté šípky), z jedného smeru ide (zelená šípka), oproti tomu smeru je prekážka.

2.2 Fitnes

Mních je ohodnotený podľa toho, ako dobre pohrabal záhradu. Hodnotenie je hodnota rozmeru záhrady mínus kombinácia niekoľkých faktorov:

- Počet nepohrabaných miest
- Či sa mních zasekol
- Počet úspešných aj neúspešných pohybov ktoré mních vykonal
- Počet opakovaní rovnakého rozloženia voľných miest(núl) na záhrade

Tieto faktory sú prenásobené číslami, aby sme im určili váhu dôležitosti. Cieľom je priradiť najvyššiu hodnotu fitnes najlepšie pohrabanej záhrade. Čím viac nepohrabaných miest, tým je fitnes horšie. Ak sa mních zasekol, zmenší sa mu fitnes oproti jedincovi, ktorý pohrabal rovnaký počet políčok ale nezasekol sa. Čím viac pohybov mních vykoná alebo skúsi vykonať, tým viac je penalizovaný. Ak sa rovnaké rozloženie núl v záhrade vyskytuje naprieč populáciou, jedince v budúcich generáciách sú za to penalizované.

2.3 Záhrada

Mních si pamätá aj záhradu, aby sme ju vedeli vypísať pomocou pomocnej funkcie *print_garden()*.

3. Genetický algoritmus

Genetický algoritmus je kombinácia nasledujúcich bodov. Na začiatok potrebujeme vytvoriť počiatočnú generáciu. Tú vytvárame náhodne pomocou funkcie *make_first_gen()*. Všetky naše generácie majú 128 jedincov, po výbere sa ich počet zmenší na 16. Dôvodom týchto čísel je voľba výberu pomocou turnaja a jeho implementácia. Vytváranie prvej generácie robíme vo funkcií main, pretože genetický algoritmus voláme dvakrát pre obidve metódy výberu. Počiatočnú generáciu ohodnotíme a opakujeme genetický cyklus (zvýraznené modrou) do momentu, kým nenájdeme perfektné riešenie alebo počet generácií presiahne 300.

Kód algoritmu:

```
def gen_algorithm(ch):
    f = open("out{}.txt".format(ch), 'w')

    for i in range(128):
        fitness(Population[0][i])

    g = 0
    while the_Monk.fitness == 0 and g < 300:
        g = evolve(g, ch, f)

    get_best()
    print("Type: ", ch, "Generations: ", g, "Best Monk in gen:", last_move)
    print_garden(the_Monk.garden)

    f.close()
def  evolve(g):
```

```
    for i in range(128):
        fitness(Population[g][i])
    elitism(g)
    print_garden(Population[g][0].garden)
    print(g)
    g_crossing(g)
    g += 1
    return g
```

Globálna premenná *Population* je dátová štruktúra, ktorá obsahuje všetky generácie. Po výbere (*elitism()* alebo *tourmanent()*) sa generácia populácií zmenší na 16 jedincov. Po skrížení (*g_crossing()*) je veľkosť populácie vždy 128, z týchto počtu sa následne znovu vyberá.

3.1 Prvá generácia

Vytvárajú sa chromozómy, ktoré obsahujú náhodné kroky (rovnaké kroky sa neopakujú), Ak sa vytvorí jedinec, ktorý neexistuje, pridá sa do hash tabuľky, ktorá obsahuje všetkých vytvorených jedincov (ich chromozómy) a pridá sa aj do populácie (nultý list *Population*).

3.2 Výber jedincov

Úlohou bolo porovnať dva typy výberu. Implementovali sme elitizmus a turnaj.

Elitizmus bol na implementáciu dosť jednoduchý. Podľa hodnoty fitnes sme zoradili generáciu a vybrali z nej prvých 16 jedincov.

```
def elitism(i):
    global Population
    Population[i] = Population[i][:16]
```

Turnaj vyberá dvojice mníchov zo začiatku a konca listu obsahujúceho generáciu. Z dvojice potom vyberá náhodne. Na 80% prežije dominantný rodič, recesívny má šancu 20%. Keďže počiatočná generácia má veľkosť 128 jedincov, tento turnaj sa zopakuje trikrát. Tým docielime výber šestnástich jedincov.

```
def tournament(i):
    global Population
    num_monks = len(Population[i])
    new_gen = []

    for k in range(3):
        new_gen.clear()

        for j in range(int(num_monks/pow(2, k+1))):
            if random.random() > 0.2:
                new_gen.append(Population[i][j])
            else:
                new_gen.append(Population[i][int(num_monks/pow(2, k+1))-j])
        Population[i] = new_gen[:]
```

3.3 Kríženie

Po výbere jedincov je ich potrebné skrížiť, aby sme dostali novú generáciu. Zo 16 jedincov potrebujeme dostať 128 nových. Tri štvrtiny (112) jedincov vytvoríme pomocou kríženia. Prvých šestnástich z nich program zmutujeme a priradí ku nim. Pomocou hash-tabuľky zisťujeme, či mních s daným chromozómom existoval, ak áno, nevytvorí rovnakého a pokračuje v hľadaní jedinečného. Kľúčovou časťou kríženia je funkcia *cross()*.

```
def cross(a, b):
    rnd = random.randint(1, len(a.chromosome)-1)
    new = np.append(a.chromosome[:rnd], b.chromosome[rnd:])

    for i in range(rnd, len(new)-1):
        where = np.where(new == new[i])
        if len(where[0]) > 1:
            new[i] = new[i] * (-1)
    return new
```

3.4 Mutovanie

Mutovanie zabezpečuje väčšiu rozmanitosť nových jedincov. Tak ako v prírode, aj my sme sa snažili urobiť mutovanie čo najviac náhodným. Myšlienkou mutovania je zmena jedného alebo viacerých génov v chromozóme. To, ako konkrétne sa mutuje má na starosti náhodné číslo *rnd1*. Podľa *rnd1* je buď nahradené jedno číslo, otočený smer daného génu alebo kombinácia týchto možností (farebne zvýraznené).

```
def mutate(a):
    while Htable.get(str(a.chromosome)):
        rnd1 = random.randint(0, 2)

        c1 = a.chromosome[random.randint(0, len(a.chromosome) - 1)]
        c1pos = np.where(a.chromosome == c1)[0][0]
        c2 = random.randint(1,x+y-1)
        c2pos = np.where(a.chromosome == c2)[0][0]

        if rnd1 == 2:
            c2 = random.randint(0,x+y-1)
            c2pos = c1pos
            mutate(a)

        if rnd1 == 1:
            c2 = random.randint(0,x+y-1) * (-1)
            c2pos = random.randint(0,x+y-1)

        if rnd1 == 0:
            a.chromosome[c1pos] = c1 * (-1)
            if a.chromosome[len(a.chromosome)-1] > x+y-1:
                a.chromosome[len(a.chromosome) - 1] = 0
            a.chromosome[len(a.chromosome)-1] +=1

    return a.chromosome
```

Výhody, nevýhody a vylepšenia

Program nie vždy nájde perfektné riešenie. Veľakrát sa k nemu len priblíži. Program je obmedzený na vývoj 300 generácií alebo ukončenie pri nájdení perfektného riešenia. Častejšie však nenájde riešenie za tento počet generácií ako nájde. Aj napriek tomu, že sme program skúsili optimalizovať nasledujúcimi vylepšeniami.

Hash-tabuľka obsahujúca už vytvorené chromozómy zabezpečuje neopakovanie rovnakých chromozómov v populácií.

Hash-tabuľka obsahujúca pozície núl v záhrade. Ak sa často opakuje rovnaká pozícia núl pri rôznych chromozómoch, tak je znižovaná hodnota fitnes. Program mal tendenciu zaseknúť sa na jedincoch s malým počtom núl na rovnakých miestach. Rozmanitosť populácie tak rýchlo klesala, ak program našiel riešenie, tak hlavne vďaka mutáciám.

MxN veľkosť záhrady je ďalším vylepšením. Používateľ si vie zadať rozmer záhrady zo vstupu. Ak je veľkosť rovná tej zo zadania (12x10), tak rozloží skaly podľa zadania. Ak je veľkosť iná, rozmiestni skaly náhodne.

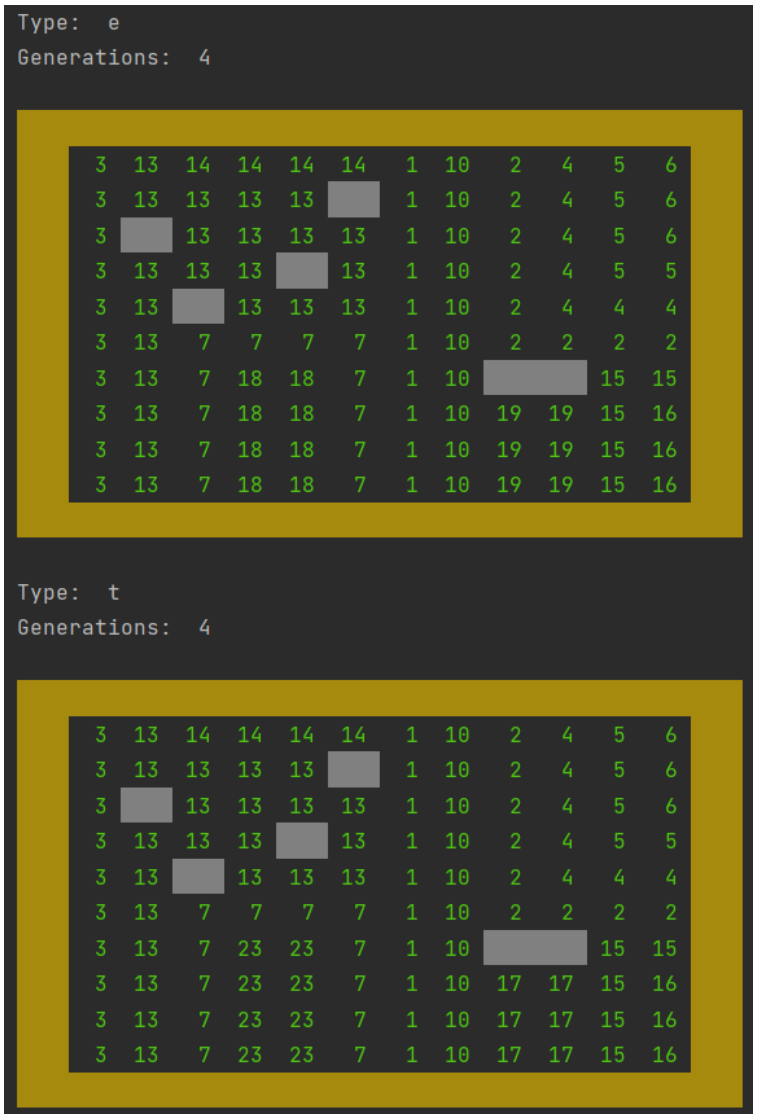
Ďalším vylepšením, ktoré sme však nerealizovali, by bolo zlepšenie prideľovania hodnoty fitnes. Napríklad uprednostňovať takých mníchov, ktorý nechávajú nepohrabané miesta pri sebe, nie oddelene na rôznych miestach záhrady. Ďalej trestať mníchov, ktorý nechávajú voľné miesta vo vnútri záhrady a kraj pohrabú, tým pádom sa nedá viac vkročiť do záhrady. Obdobne by sme mohli znevýhodňovať aj tých mníchov, ktorí začínajú hrabať na krajoch a tým zamedzia prístupu do záhrady z daného kraja.

Okrem vylepšenia fitnes vidíme nedokonalosť aj pri krížení. Taktiež naše riešenie nie je optimalizované, čo sa týka časovej náročnosti.

Testovanie

Ako bolo spomínané v zadaní, testovali sme pre rôzne metódy výberu jedincov. Okrem toho sme testovali dovtedy, dokým sme v prvých 300 generáciách nenašli perfektné riešenia aj pre metódu turnaj, aj elitizmus.

Výsledkom programu je takýto výpis. „Type“ hovorí o použitej metóde výberu. „Generation“, teda generácia, označuje generáciu do ktorej sa populácia vyvinula.



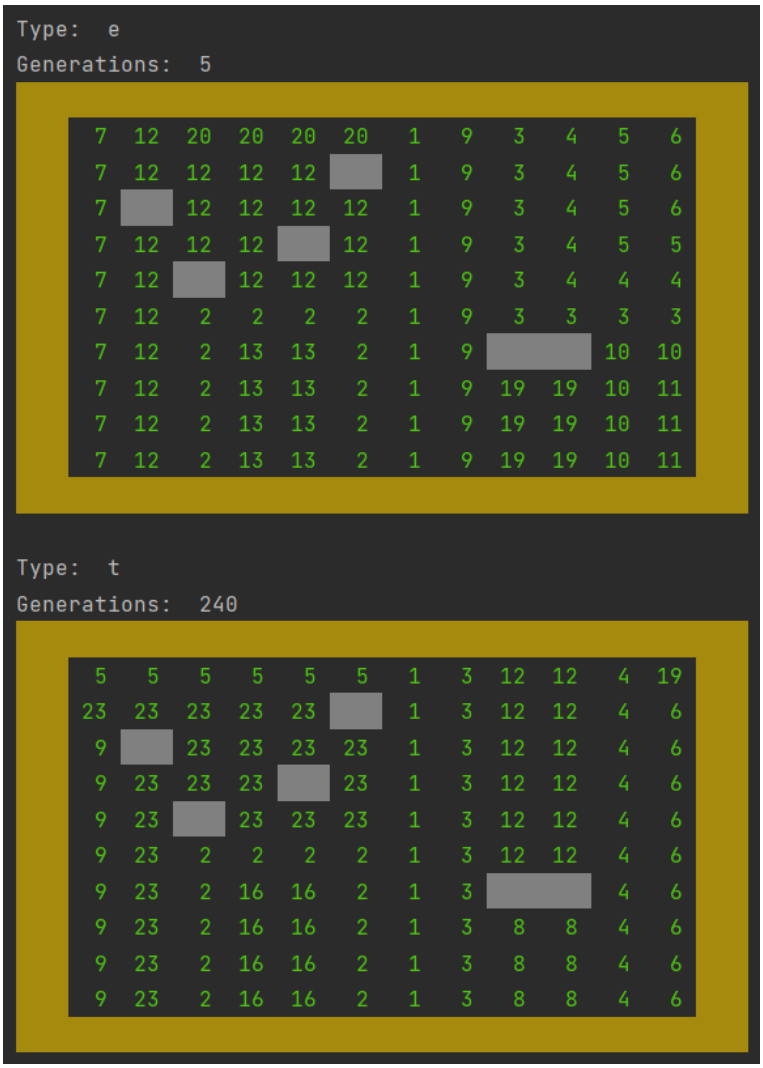
Ako vidíme, záhrada je pohrabaná totožne, no poradie niektorých génov bolo na inom mieste v chromozóme.

Výsledky

Výsledky testovania záležali hlavne na prvej resp. nultej generácií a náhodnosti mutovania.

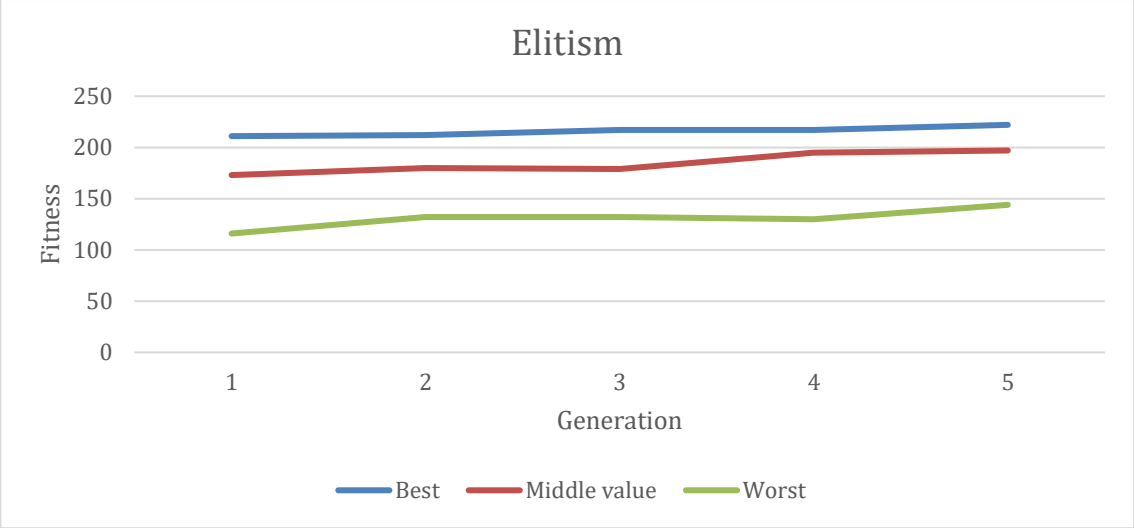
Nájdenie riešenia

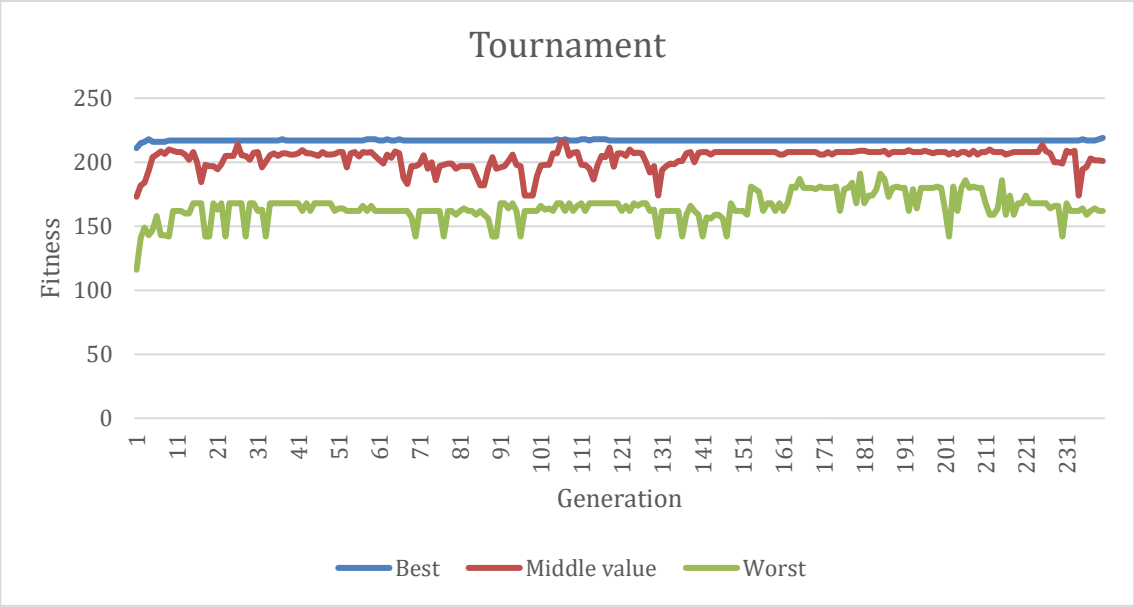
V niekoľkých prípadoch program našiel riešenie aj pre metódu výberu elitizmom aj turnajom. V týchto prípadoch bola vo väčšine rýchlejšia metóda elitizmus.



Prvá metóda našla už v priebehu piatich generácií (640 jedincov v populácií). Druhá metóda až v 240 generácií (30 720 jedincov v populácií). Štýl pohrabania je podobný ale nie úplne totožný, líšia sa aj v počte použitých génov. Druhá metóda využila všetkých 22 génov na pohyb (čísla od 1 - 23) síce nie všetky viedli k ťahu po záhrade. Prvý metóda skončila devätnástym génom (ťah 20 na záhrade).

Ku týmto populáciám odpovedajú nasledujúce grafy:





Nenájdenie riešenia

V prípade, že program nenájde riešenie počas 300 generácií, skončí a vypíše toho s najlepším fitnes z celej populácie, teda toho, ktorý sa najbližšie priblížil ku riešeniu.

Počas testovania sme testovali aj s vylepšením aj bez vylepšenia na zmenšenie fitnes pri mníchoch, ktorí majú záhrady s rovnakými pozíciami núl. Keď bolo toto vylepšenie zapnuté, grafy, konkrétne krivka najlepšieho skóre fitnes sa držala takmer v rovine. Pri vypnutí bolo poznať účinnosť tohto vylepšenia z grafov. Program sa nespoliehal, že niekedy pomocou mutácie trafí výsledok ale jednotlivý členovia generácie, ktorí boli viac jedineční, boli uprednostňovaní. Rozdiel môžeme pozorovať na grafoch.

Záhrady pre vypnuté vylepšenie:

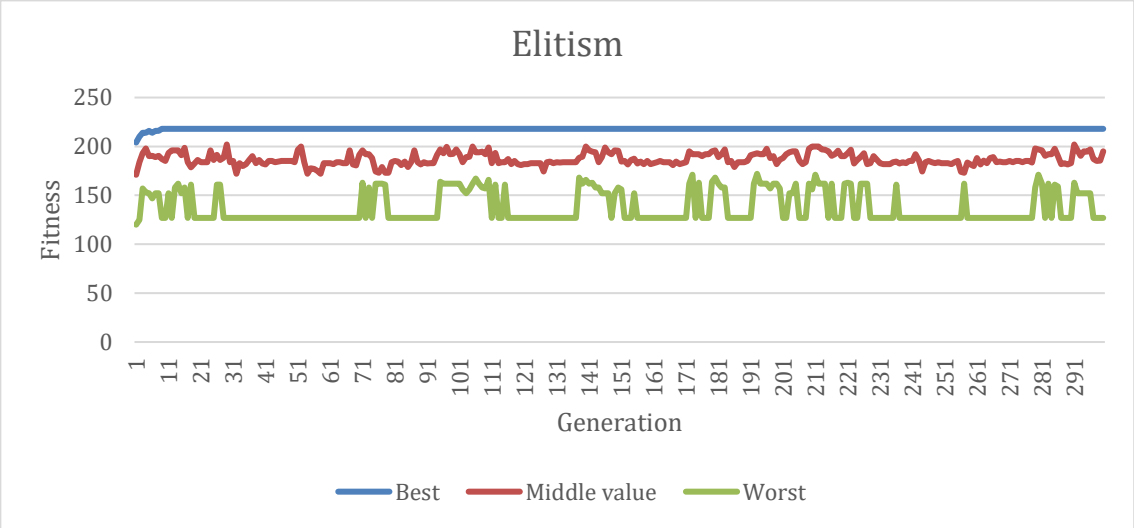
```
Type:  e
Generations:  300
Best Monk in generation:  8
```

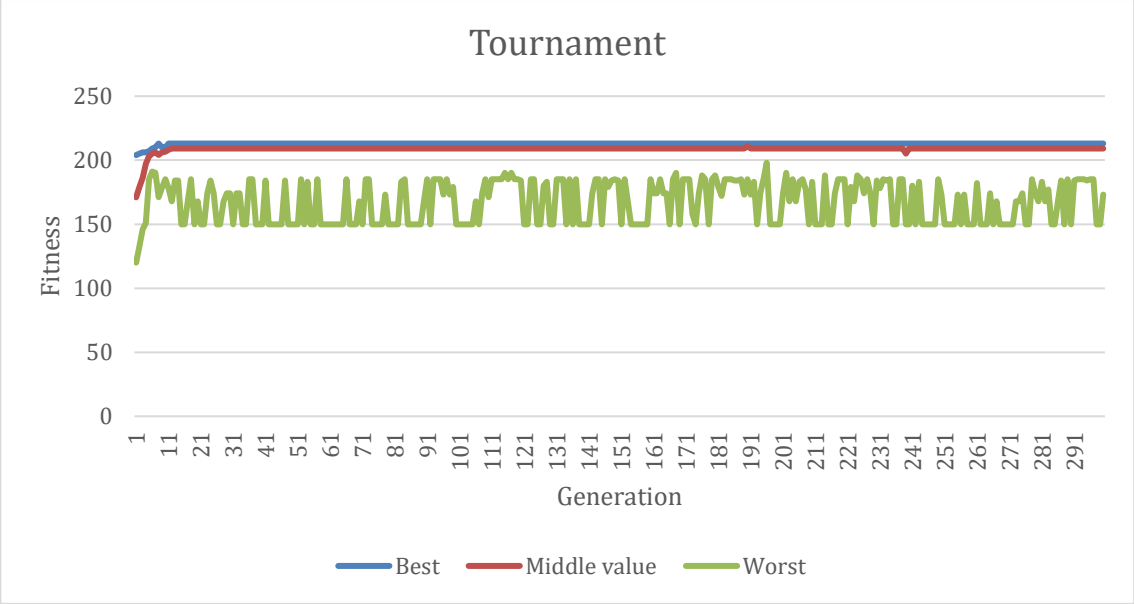
5	5	5	5	5	5	2	19	1	3	4	12
8	8	8	8	8		2	19	1	3	4	12
0		8	8	8	8	2	19	1	3	4	12
8	8	8	8		8	2	19	1	3	4	4
14	14		8	8	8	2	19	1	3	3	3
16	14	6	6	6	6	2	19	1	1	1	1
16	14	6	11	11	6	2	19			13	13
16	14	6	11	11	6	2	19	21	21	13	15
16	14	6	11	11	6	2	19	21	21	13	15
16	14	6	11	11	6	2	19	21	21	13	15

```
Type:  t
Generations:  300
Best Monk in generation:  7
```

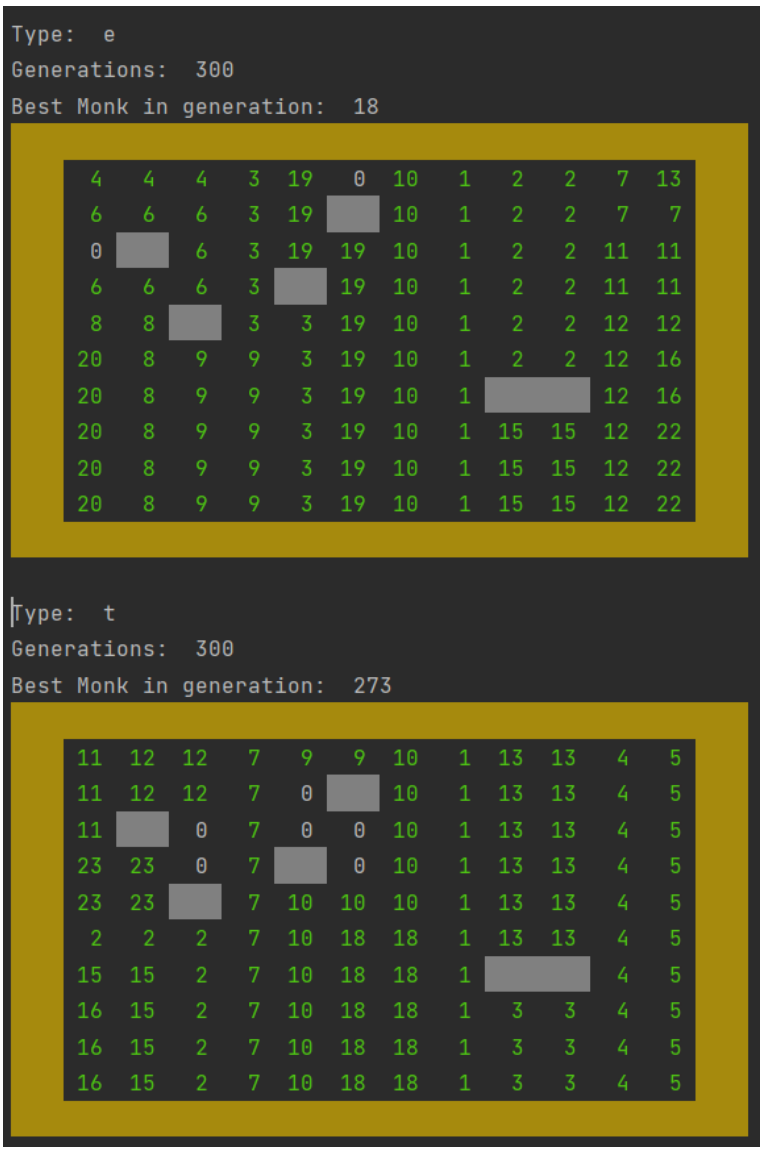
8	8	8	8	8	8	1	2	12	12	3	4
5	5	5	5	5		1	2	12	12	3	4
6		0	0	5	5	1	2	12	12	3	4
6	0	0	0		5	1	2	12	12	3	4
6	0		13	13	5	1	2	12	12	3	4
6	19	19	13	13	5	1	2	12	12	3	4
6	19	19	13	13	5	1	2			3	4
6	19	19	13	13	5	1	2	16	16	3	4
6	19	19	13	13	5	1	2	16	16	3	4
6	19	19	13	13	5	1	2	16	16	3	4

Grafy pre vypnuté vylepšenie:

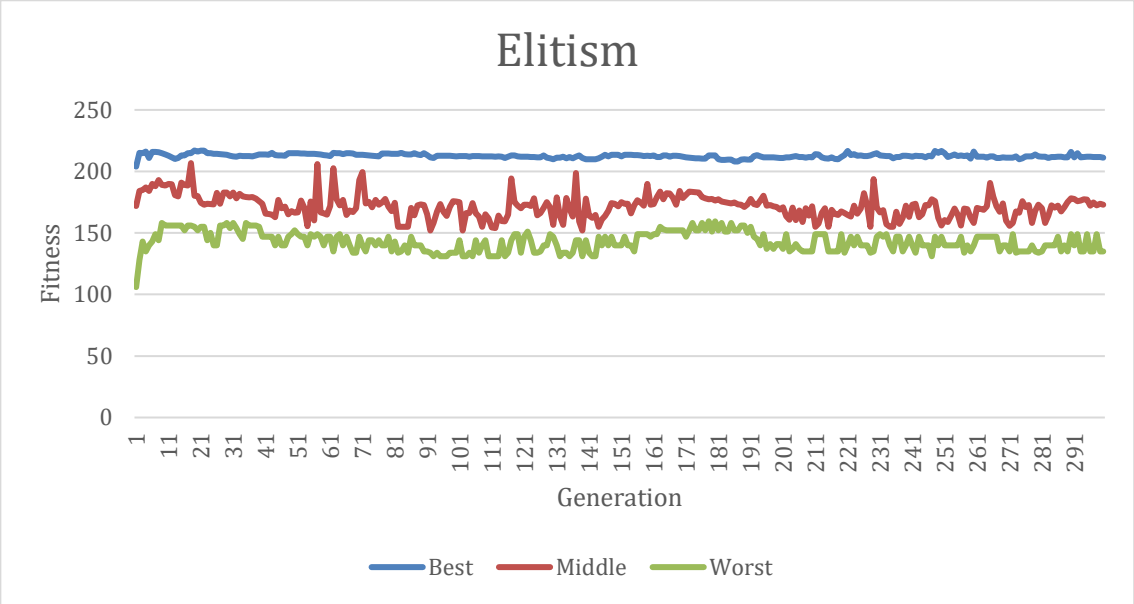


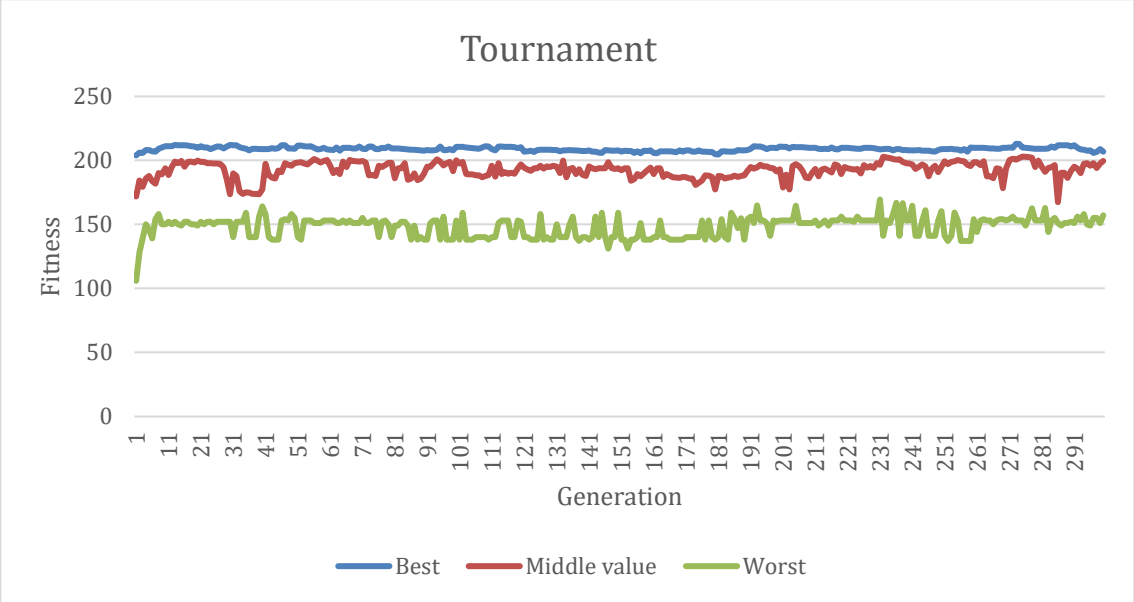


Záhrady pre zapnuté vylepšenie:



Grafy pre vypnuté vylepšenie:



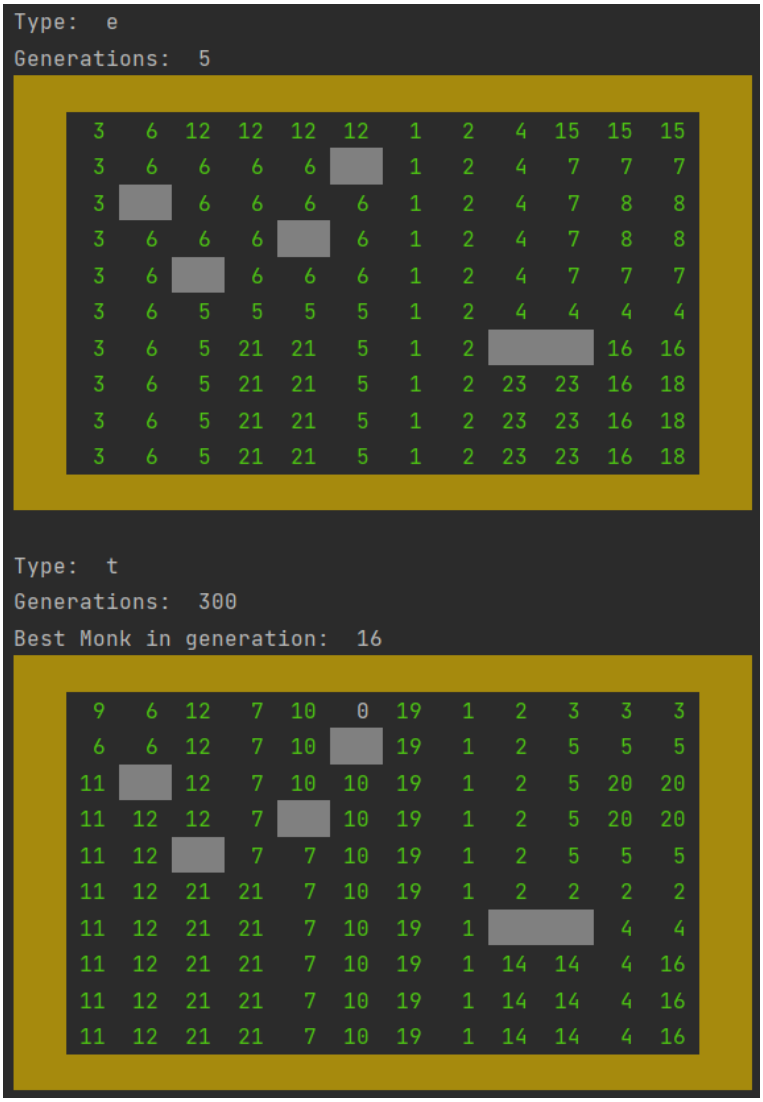


Nájdenie len v jednom prípade

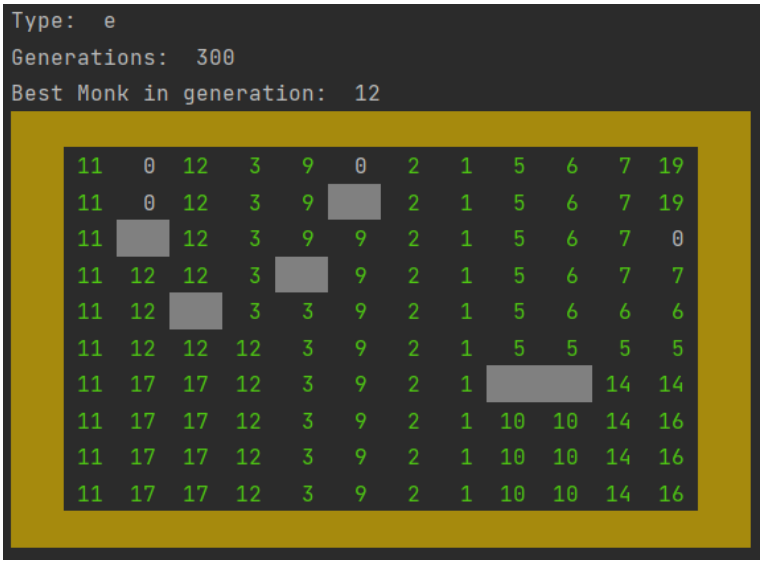
Okrem nájdenia resp. nenájdenia perfektného riešenia sa stávalo aj to, že len jedna metóda výberu našla riešenie pri totožnej prvej generácií. Túto skutočnosť pripisujeme faktu, že dané metódy sú odlišné a každá má svoje plusy aj mínusy, okrem toho to mohlo byť ovplyvnené náhodou pri výbere náhodných čísle počas mutovania génu.

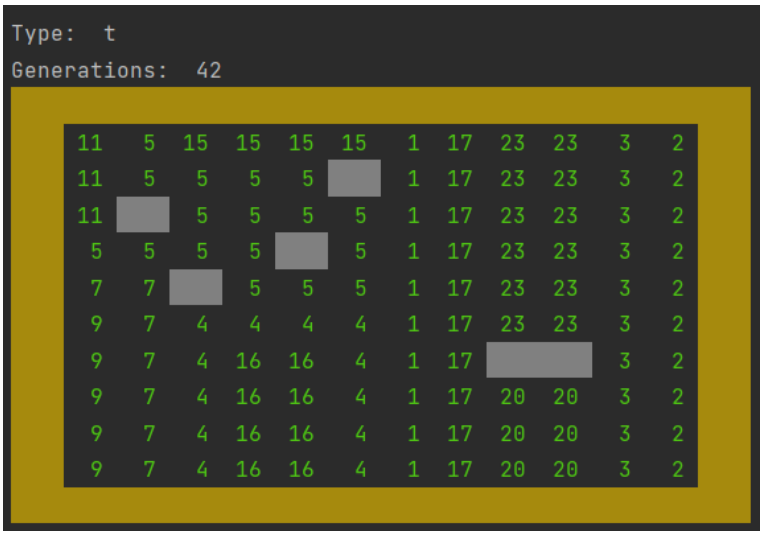
Prípady, kedy sa našlo perfektné riešenie vyzerajú takto:

Elitizmus našiel, turnaj nie.



Turnaj našiel, elitizmus nie.

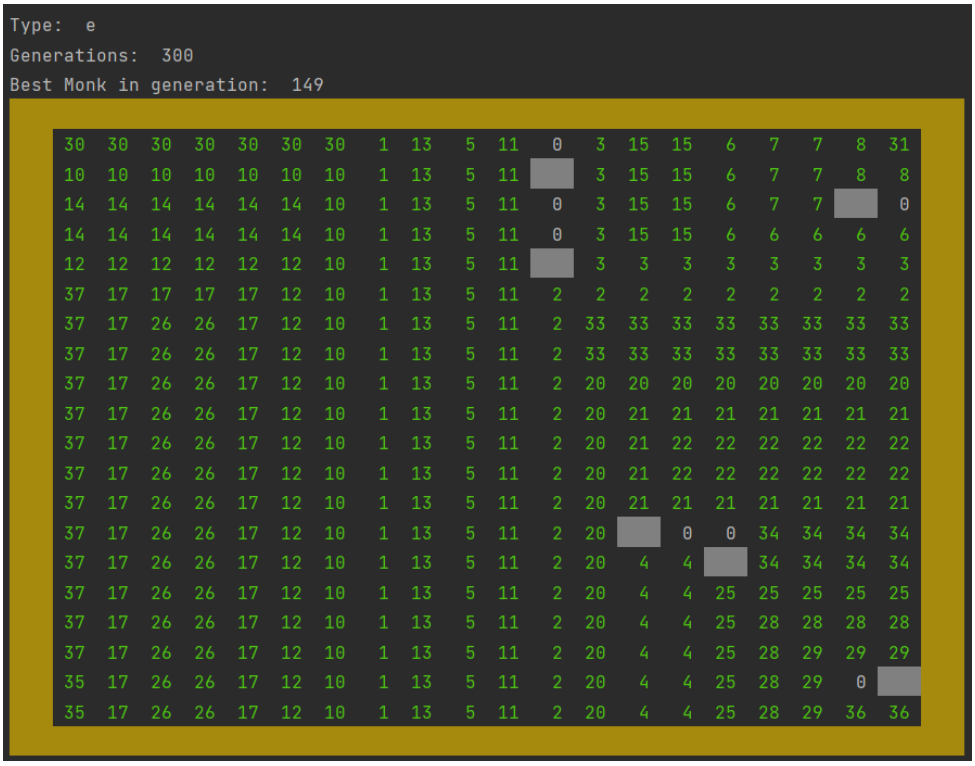




Väčší rozmer ako zadanie

Testovali sme rozmer 20x20. Ku riešeniu sme sa vedeli len priblížiť.

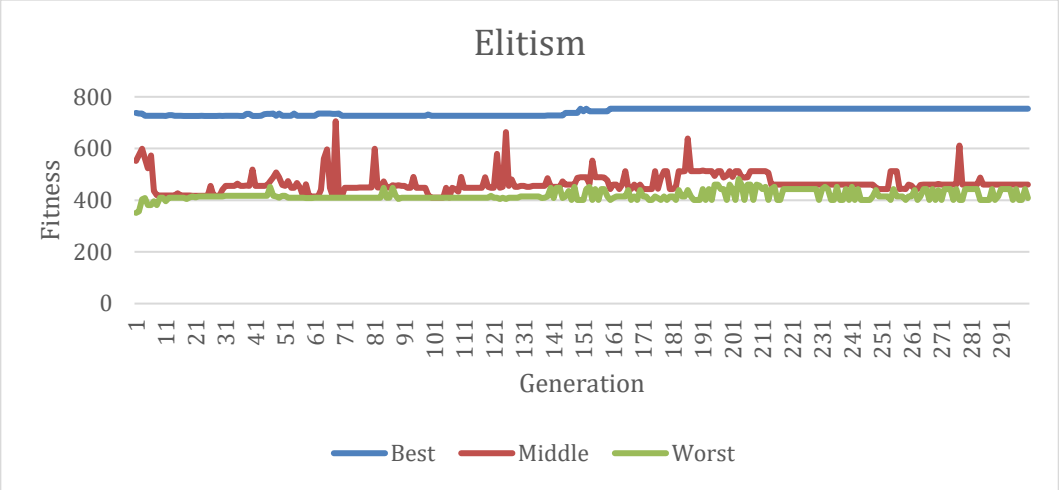
Záhrady:

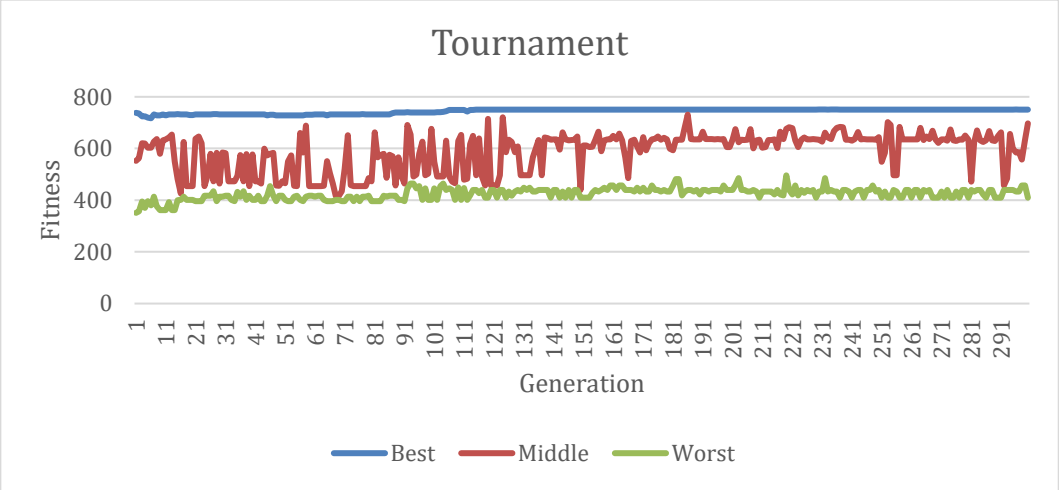


```
Type:  t
Generations:  300
Best Monk in generation:  233
```

15	15	15	15	15	15	15	1	33	5	11	0	3	9	36	36	20	20	0	41
10	10	10	10	10	10	10	1	33	5	11		3	9	36	36	20	20	0	41
22	22	19	19	14	14	10	1	33	5	11	0	3	9	36	36	20	20		41
24	22	19	19	14	14	10	1	33	5	11	0	3	9	9	9	9	9	9	9
24	22	19	19	14	14	10	1	33	5	11		3	3	3	3	3	3	3	3
24	22	19	19	14	14	10	1	33	5	11	2	2	2	2	2	2	2	2	2
24	22	19	19	14	14	10	1	33	5	11	2	16	16	16	16	16	16	16	16
24	22	19	19	14	14	10	1	33	5	11	2	16	18	18	18	18	18	18	18
24	22	19	19	14	14	10	1	33	5	11	2	16	18	21	21	21	21	21	21
24	22	19	19	14	14	10	1	33	5	11	2	16	18	21	23	23	23	23	23
24	22	19	19	14	14	10	1	33	5	11	2	16	18	21	23	23	23	23	23
24	22	19	19	14	14	10	1	33	5	11	2	16	18	21	21	21	21	21	21
24	22	19	19	14	14	10	1	33	5	11	2	16	18	18	18	18	18	18	18
24	22	19	19	14	14	10	1	33	5	11	2	16		0	0	28	28	28	28
24	22	19	19	14	14	10	1	33	5	11	2	16	4	4		28	28	28	28
24	22	19	19	14	14	10	1	33	5	11	2	16	4	4	6	6	6	6	6
24	22	19	19	14	14	10	1	33	5	11	2	16	4	4	6	7	7	7	7
24	22	19	19	14	14	10	1	33	5	11	2	16	4	4	6	7	0	8	8
24	22	19	19	14	14	10	1	33	5	11	2	16	4	4	6	7	0	8	
24	22	19	19	14	14	10	1	33	5	11	2	16	4	4	6	7	0	8	32

Grafy:

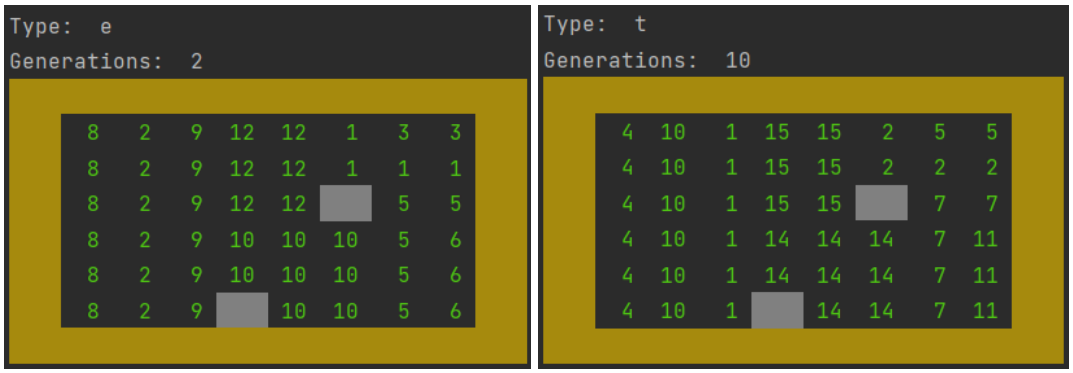




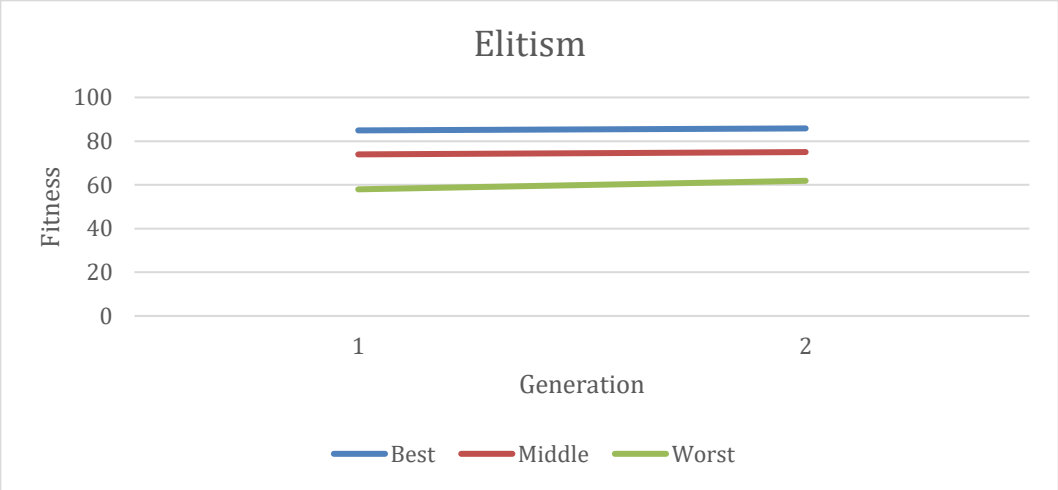
Menší rozmer ako zadanie

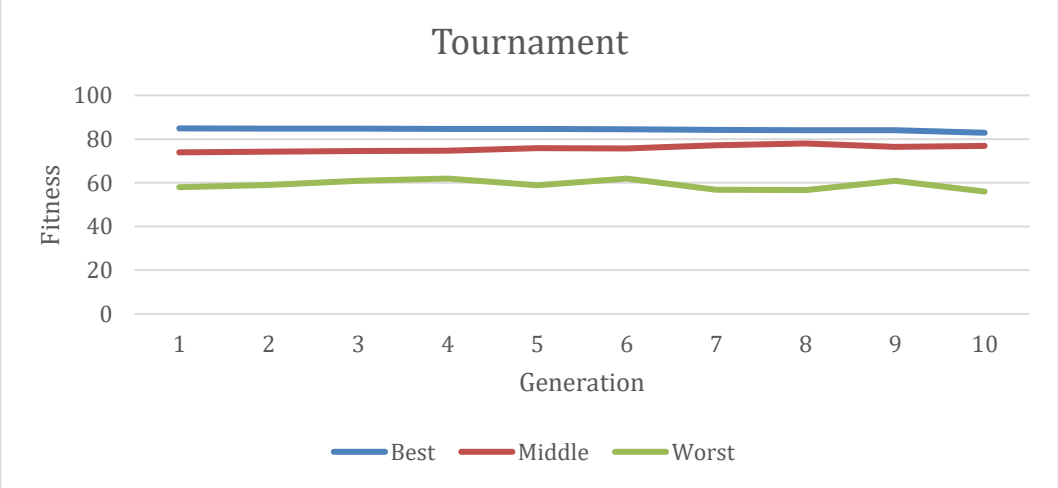
Pri zmenšovaní záhrady klesala aj náročnosť na nájdenie perfektného riešenia. Problémom však bolo nastavenie hodnoty fitnes. Pri elitizme sa program dostal do bodu, kedy mu nevadilo, že sa zasekol. Fitnes bolo lepšie ako keby mal voľné miesto a tak pokračoval v hľadaní. Metóda turnaj sa je viac rozmanitá, tento problém pri nej nastal ojedinele. Keď však program našiel riešenie, vyzeralo to nasledovne.

Záhrady:



Grafy:





Zhodnotenie

Ako sme už spomínali v kapitole Výhody, nevýhody a vylepšenia, program nie je úplne optimalizovaný. Nenájde vždy perfektné riešenie, ale vždy sa k nemu priblíži.

Testovaním sme preukázali rôzne stavy, ktoré môžu nastať, rovnako aj opísali odôvodnenie týchto stavov.

Úlohu sme splnili, perfektné riešenie program našiel v niekoľkých prípadoch, viac v kapitole Výsledky. Nájdenie riešenie závisí hlavne od dolaďovania fitnes fukncie a implementácie ďalších vylepšení. V priebehu písania programu bola fitnes funkcia niekoľkokrát pozmenená. Momentálne nastavenie je určené pre obidve metódy výberu. Pri inom nastavení bol elitizmus oveľa rýchlejší v nájdení perfektného riešenia, ale metóda turnaj sa ledva priblížila ku riešeniu.

Časti programu, ktoré nie sú opísané v dokumentácií sú okomentované v zdrojovom kóde *main.py*.

Program je programovaný v IDE PyCharm, spúšťanie programu tu umožňuje farebný výpis, ktorý nefunguje v cmd.