**A close up of a logo

Description automatically generated**

Umelá inteligencia

Zen záhrada - Simulované žihanie

Jakub Šimko

AIS ID: 103146

2021/2022

Cvičiaci: Ing. Ivan Kapustík

Čas cvičení: Streda 13:00

Obsah

[1. Zadanie úlohy 4](#_Toc88302651)

[2. Úvod 6](#_Toc88302652)

[3. Gény 7](#_Toc88302653)

[3.1. Štartovacie pozície 7](#_Toc88302654)

[3.2. Rozhodovanie pri kolízii 8](#_Toc88302655)

[3.3. Maximálny počet génov 9](#_Toc88302656)

[4. Mních / Záhradník 9](#_Toc88302657)

[4.1. Získanie fitness – hrabanie záhrady 10](#_Toc88302658)

[4.1.1. Cyklus hrabania 11](#_Toc88302659)

[5. Hľadanie susedov 12](#_Toc88302660)

[5.1. Prvý variant 12](#_Toc88302661)

[5.1.1. Prehodenie pozícii dvoch génov 13](#_Toc88302662)

[5.1.2. Inverzia rozhodovania dvoch náhodných génov 13](#_Toc88302663)

[5.2. Druhý variant 14](#_Toc88302664)

[6. Simulované žihanie 15](#_Toc88302665)

[6.1. Úprava kvôli testovaniu 16](#_Toc88302666)

[7. Testovanie a výsledky pre rôzne nastavenia 16](#_Toc88302667)

[7.1. Testovanie – nastavenie 1 17](#_Toc88302668)

[7.1.1. Testovanie nastavenia s náhodnými chromozónmi 17](#_Toc88302669)

[7.1.2. Testovanie nastavenia s určenými chromozónmi 17](#_Toc88302670)

[7.2. Testovanie – nastavenie 2 18](#_Toc88302671)

[7.2.1. Testovanie nastavenia s náhodnými chromozónmi 18](#_Toc88302672)

[7.2.2. Testovanie nastavenia s určenými chromozónmi 19](#_Toc88302673)

[7.3. Porovnanie nastavení 19](#_Toc88302674)

[7.3.1. Priebeh fitness 20](#_Toc88302675)

[8. Záver 21](#_Toc88302676)

[8.1. Dolaďovanie a zlepšovanie 21](#_Toc88302677)

Obrázky

[Obrázok 1: Reprezentácia štartovacích pozícii 7](#_Toc88302678)

[Obrázok 2: Pohyb pri štarte na pozíciach 1, 9 a 6 (nezadefinované správanie pri kolízii) 7](#_Toc88302679)

[Obrázok 3: Gén je záporny - mních odbočil do ľava 8](#_Toc88302680)

[Obrázok 4: Gén nie je záporny - mních odbočil do prava 8](#_Toc88302681)

[Obrázok 5: Funkcia na výpočet maximálneho počtu génov 9](#_Toc88302682)

[Obrázok 6: Trieda záhradnika 9](#_Toc88302683)

[Obrázok 7: Trieda políčka - uchováva informácie o pohybe a pozícií 9](#_Toc88302684)

[Obrázok 8: Implementácia hrabania na základe chromozómu - skrytý cyklus hrabania 10](#_Toc88302685)

[Obrázok 9: Určenie pozície prvého políčka a smeru pohybu na základe štartovacej pozície 10](#_Toc88302686)

[Obrázok 10: Cyklus hrabania vo funkcii rake\_garden() 11](#_Toc88302687)

[Obrázok 11: Návratové hodnoty funkcie rake\_garden() 11](#_Toc88302688)

[Obrázok 12:Nastavenie hľadania susedov 12](#_Toc88302689)

[Obrázok 13: Implementácia prvého variatnu - a), b) 12](#_Toc88302690)

[Obrázok 14: Ukážka realizácie prvého variantu - a) 13](#_Toc88302691)

[Obrázok 15: Prvá ukážka realizácie prvého variantu – b) 13](#_Toc88302692)

[Obrázok 16: Druhá ukážka realizácie prvého variantu – b) 13](#_Toc88302693)

[Obrázok 17: Ukážka realizácie druhého variantu 14](#_Toc88302694)

[Obrázok 18: Aktuálna implementácia druhé variantu 14](#_Toc88302695)

[Obrázok 19: Príklad nastavenia simulovaného žíhania 15](#_Toc88302696)

[Obrázok 20: Implementácia simulovaného žíhania 15](#_Toc88302697)

[Obrázok 21: Nastavenie 1 pre simulované žíhanie 16](#_Toc88302698)

[Obrázok 22: Výsledok testovania s náhodnými chromozónmi – nastavenie 1 17](#_Toc88302699)

[Obrázok 23: Určené začiatočné/prvé chromozómy 17](#_Toc88302700)

[Obrázok 24: Výsledok testovania so zadanými chromozónmi pre nastavenie 1 – pokus č. 1 17](#_Toc88302701)

[Obrázok 25: Výsledok testovania so zadanými chromozónmi pre nastavenie 1 - pokus č. 2 18](#_Toc88302702)

[Obrázok 26: nastavenie 2 18](#_Toc88302703)

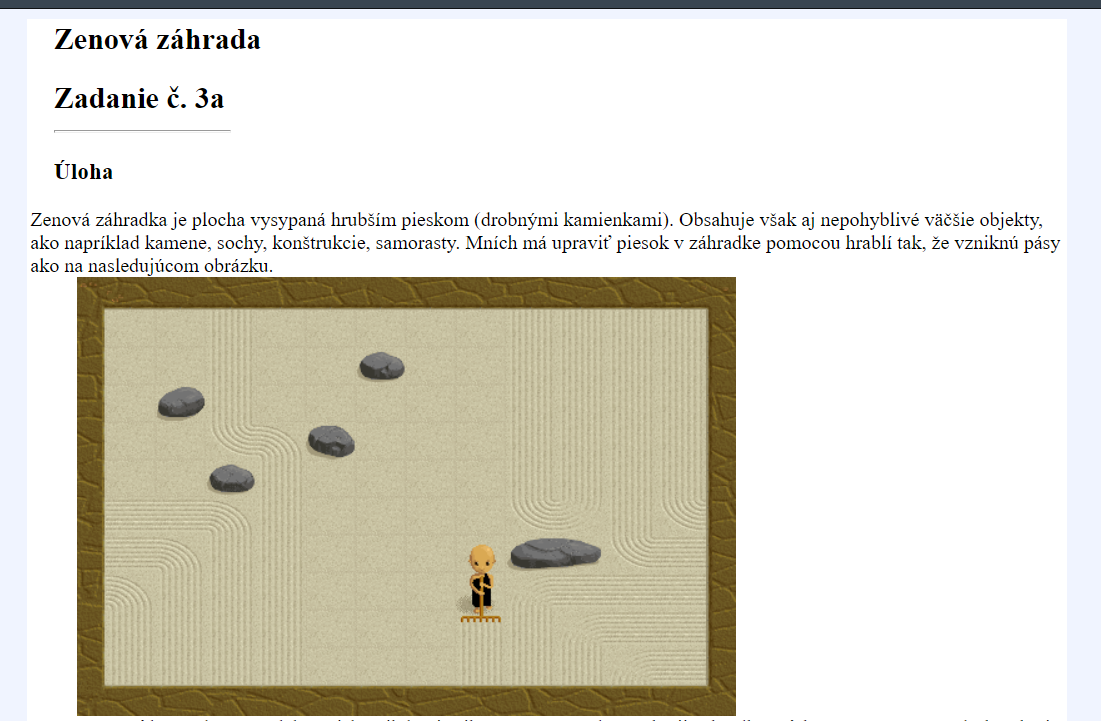
[Obrázok 27: Výsledok testovania s náhodnými zač. chromozónmi – nastavenie 2 18](#_Toc88302704)

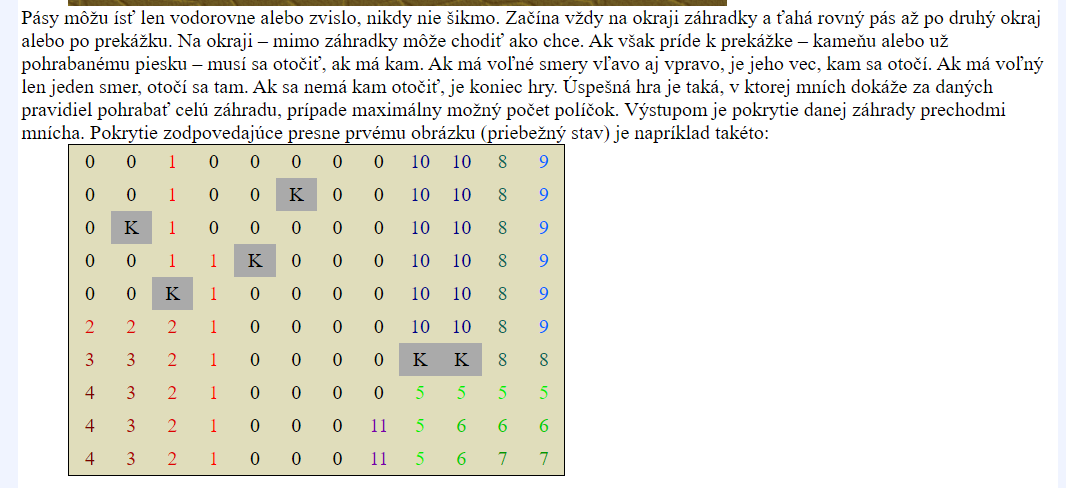
[Obrázok 28: Výsledok testovania so zadanými chromozónmi pre nastavenie 2 – pokus č. 1 19](#_Toc88302705)

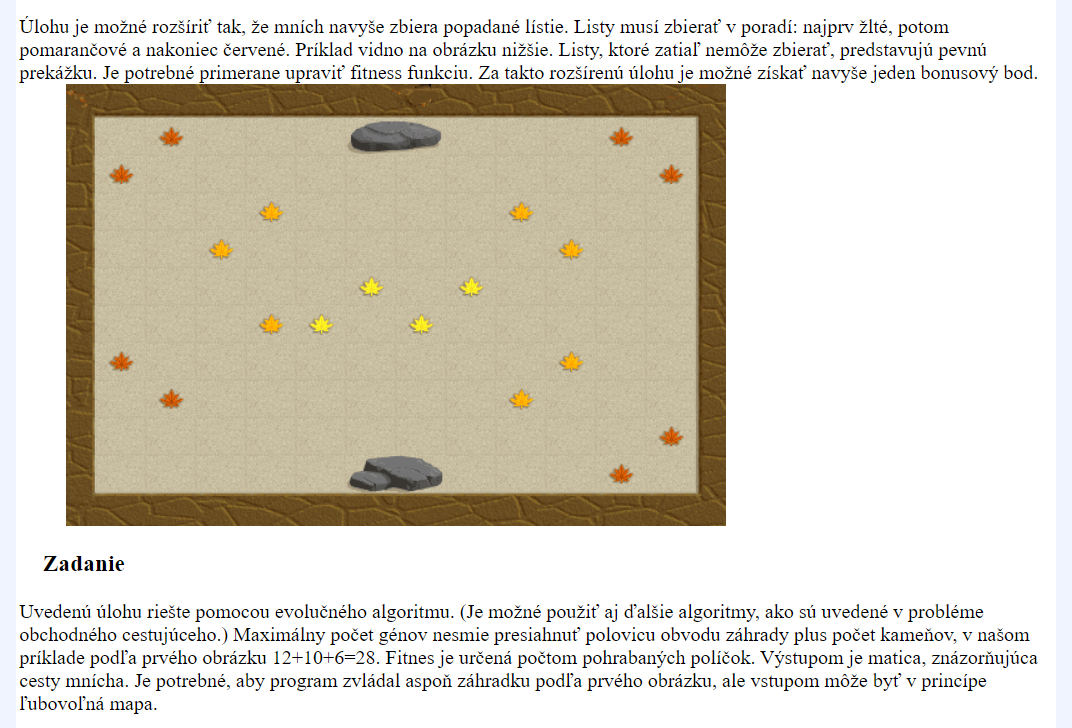
[Obrázok 29: Výsledok testovania so zadanými chromozónmi pre nastavenie 2 - pokus č. 2 19](#_Toc88302706)

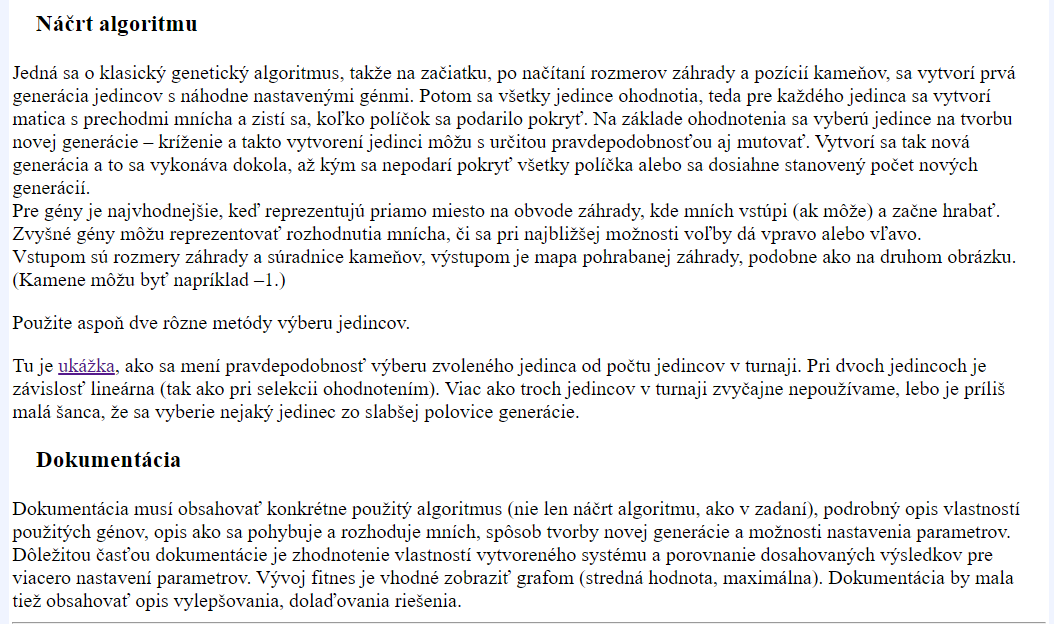
[Obrázok 30: Výsledok testovania so zadanými chromozónmi pre nastavenie 2 - pokus č. 3 19](#_Toc88302707)

# Zadanie úlohy









# Úvod

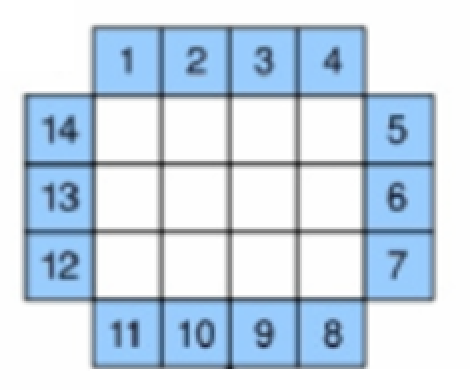
Úlohou je implementovať funkčné riešenie pre problém zen záhrady využitím algoritmu simulovaného žíhania. Daný problém som rozdelil na viacero menších častí:

1. Ako zadefinovať gény
   1. Štartovacia pozícia
   2. Rozhodovanie
2. Ako odsimulovať hrabanie daného mnícha
3. Ako získať ohodnotenie daného pohrabania
4. Ako hľadať susedov
5. Implementácia simulovaného žíhania
6. Správne nastavenia simulovaného žihania a hľadania susedov

# Gény

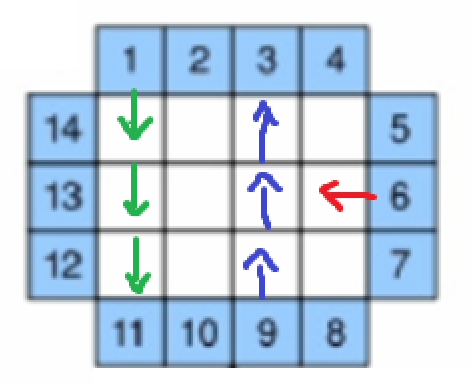
Za najzložitejšiu časť tohto zadania považujem tú v ktorej som musel vymyslieť ako reprezentovať jednotlivé gény. Po dlhšom uvažovaní som dospel k tomu, že gén musí v sebe niesť informáciu o štartovacej pozícií a informáciu o tom ako sa mních rozhodne ak dôjde ku kolízií.

## Štartovacie pozície



Obrázok 1: Reprezentácia štartovacích pozícii

Štartovacie pozície som sa rozhodol reprezentovať tak ako sú znázornené na obrázku č. 1. To znamená, že štartovacie políčko číslo 1 označuje začiatok hrabania na políčku s indexmi 0,0 a mních v prípade tejto začiatočnej pozície hrabe smerom dole. Štartovacia pozícia 14 reprezentuje rovnaké začiatočné políčko ale hrabanie mnícha bude prebiehať smerom do prava. To vlastne znamená, že v prípade štartovacej pozície č. 1 vchádza mních do záhrady zhora v prípade č. 14 vchádza do záhrady z ľava.



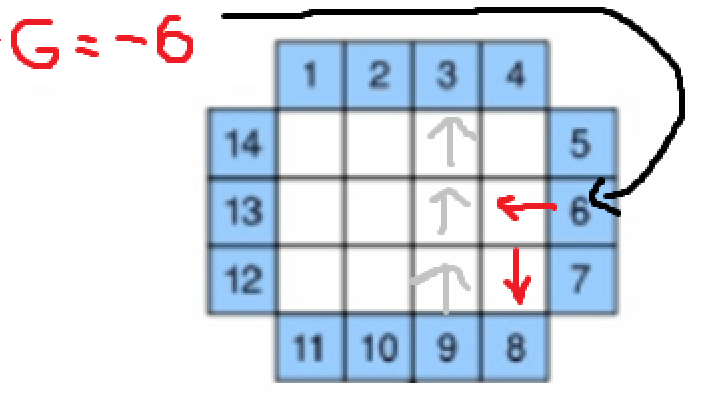
Obrázok 2: Pohyb pri štarte na pozíciach 1, 9 a 6 (nezadefinované správanie pri kolízii)

Zadefinovaním štartovacích pozícií je vyriešená prvá čast implementácie génov.

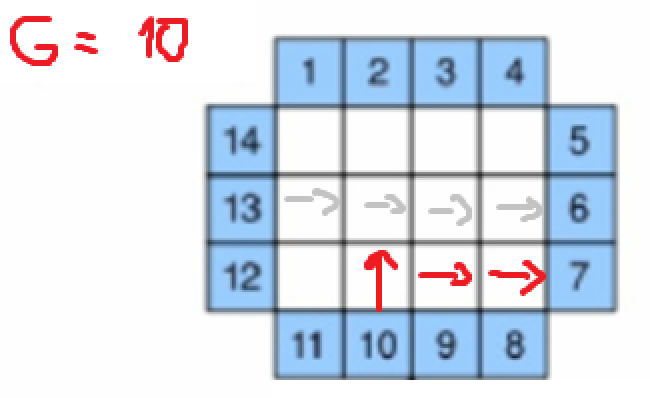
## Rozhodovanie pri kolízii

Ako ďalšie je nutné vymyslieť ako bude v danom géne uložená informácií o tom ako sa pri kolízií mních rozhodne. Ak dôjde ku kolízii a mních ma na výber dva nové smery hrabania, tak si vyberie podľa génu. Ak je na výber len jedna možnosť tak sa pre ňu rozhodne – nezáleži na géne. V prípade kedy neni na výber žiadny nový smer tak sa mních zasekol.

Vzhľadom na to, že chromozóm je pole celých čísel (int), tak som sa rozhodol v jednotlivých génoch reprezentovať informáciu o rozhodnutí nového smeru pri kolízii znamienkami + a -. Keď dôjde ku kolízií a daný gén je záporny tak z pohľadu mnícha, začne hrabať smerom vľavo v opačnom prípade začína hrabať smerom doprava. Logika tohto správania je implementovaná vo funkcii **decide\_direction().**



Obrázok 3: Gén je záporny - mních odbočil do ľava



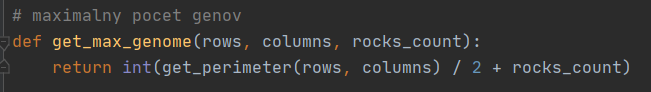
Obrázok 4: Gén nie je záporny - mních odbočil do prava

Príklad reprezentácie náhodného chromozómu pre záhradu zo zadania:

[37, -15, 31, -12, -36, 16, -35, -8, 26, -11, -38, 7, 34, -2, -18, -17, 28, -21, -25, -40, -10, 14, 19, 33, 4, 27, -42, -3]

## Maximálny počet génov

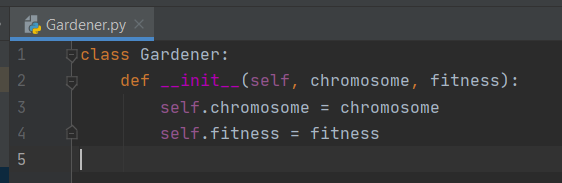
Maximálny počet génov je určený v zadaní ako polovica obvodu záhrady + počet kameňov.



Obrázok 5: Funkcia na výpočet maximálneho počtu génov

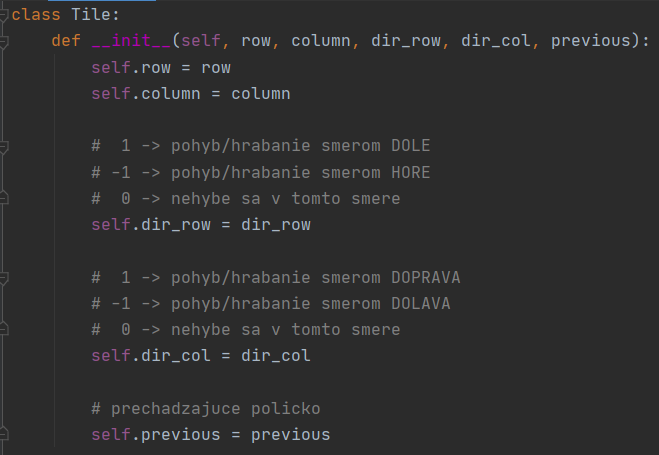
# Mních / Záhradník

Záhradník je reprezentovaný chromozómom zo sekcie 3 a ohodnotením – fitness.



Obrázok 6: Trieda záhradnika

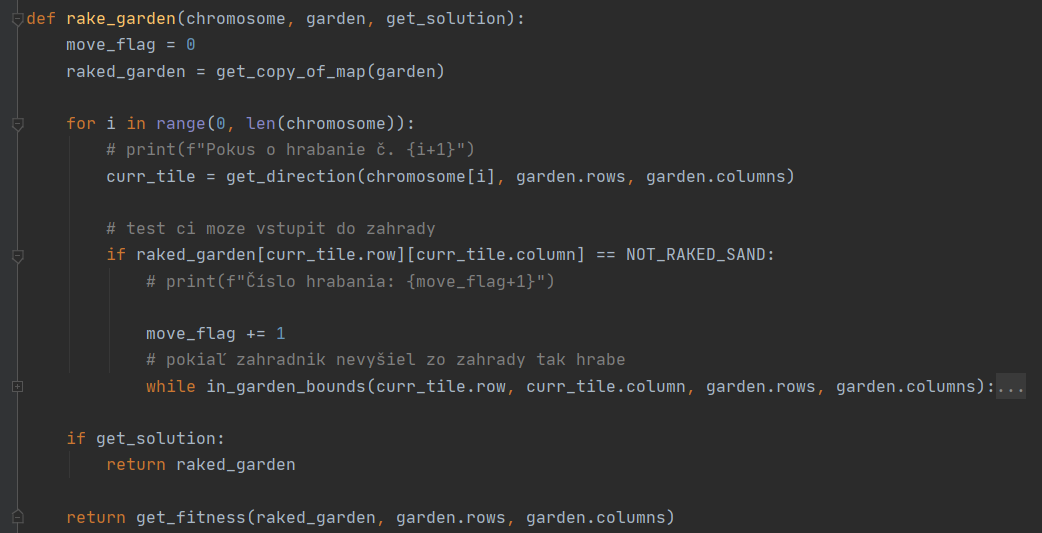
Mních v sebe neuchováva informácií o tom, ktorým smerom aktuálne hrabe alebo na akom políčku (súradniciach) sa aktuálne nachádza. Tieto informácie sú uložené v triede Tile, ktorá je využívaná len pri hrabaní vo funkcii **rake\_garden()**.



Obrázok 7: Trieda políčka - uchováva informácie o pohybe a pozícií

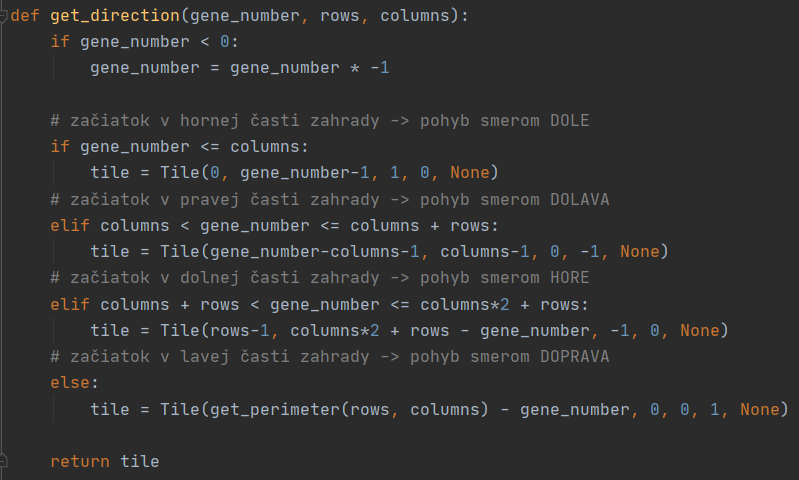
## Získanie fitness – hrabanie záhrady

Na to aby sme dokázali určiť fitness mnícha tak je nutné aby nad záhradou vykonal hrabanie – musíme odsimulovať jeho riešnie. Na to slúži funkcia **rake\_garden()**. Na začiatku funkcie sa vytvorí kópia záhrady aby sa hrabanie neuložilo do pôvodnej záhrady. Následné sa pre každý gén vykoná hrabanie.



Obrázok 8: Implementácia hrabania na základe chromozómu - skrytý cyklus hrabania

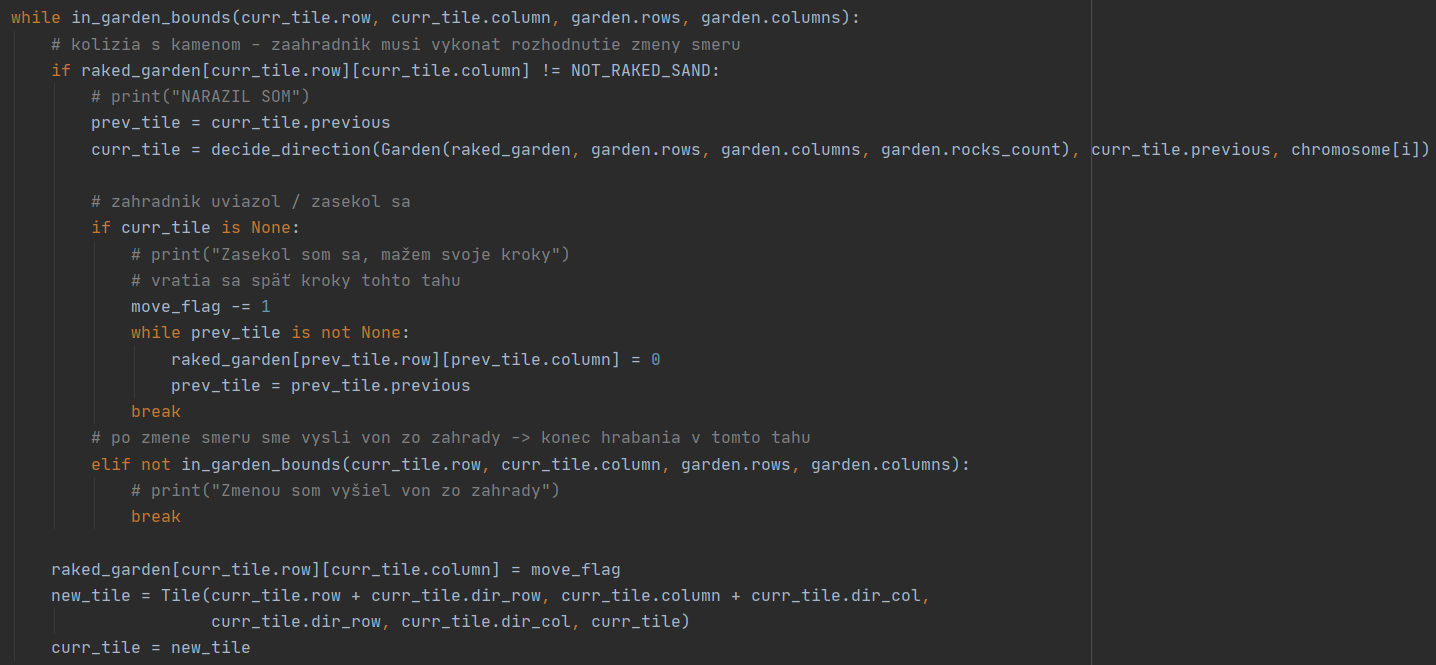
Na základe začiatočnej pozície získa zač. políčko a smer hrabania. Následne sa skontroluje vo funkcii **rake\_garden()** či je možné vstupiť do záhrady na toto políčko.



Obrázok 9: Určenie pozície prvého políčka a smeru pohybu na základe štartovacej pozície

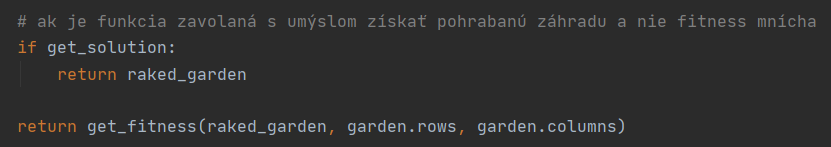
### Cyklus hrabania

Komentáre v kóde jednoducho vysvetľujú logiku hrabania. Najdôležitejšou funkciou tohto cyklu je funkcia **decide\_direction()**, ktorej správanie je vysvetlené v sekcii 3.2.



Obrázok 10: Cyklus hrabania vo funkcii rake\_garden()

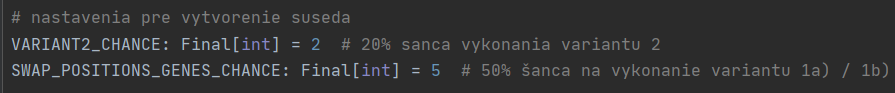
Na konci funkcia vráti fitness daného mnícha alebo pohrabanú záhradu. Záleží od tretieho argumentu funkcie (*get\_solution*).



Obrázok 11: Návratové hodnoty funkcie rake\_garden()

# Hľadanie susedov

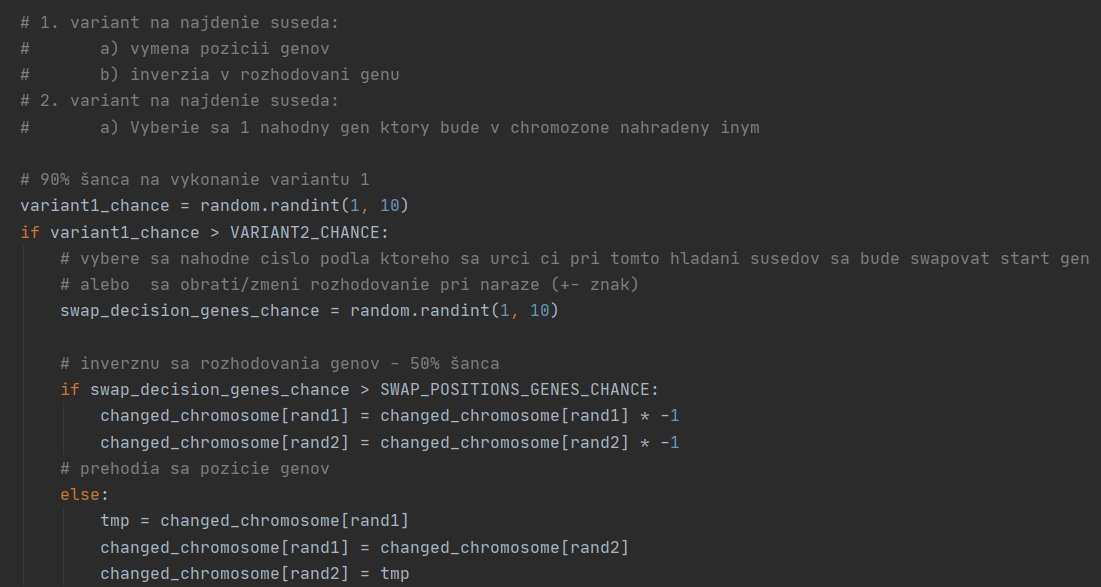
V rámci hľadania susedov som postupne vylepšoval funkciu *get\_neighbour()*. V programe sú implementované tri spôsoby hľadania suseda.



Obrázok 12:Nastavenie hľadania susedov

## Prvý variant

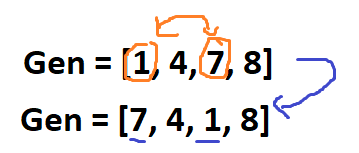
Šanca na vykonnie prvého variantu oproti variantu dva je 80%. Šanca na vykonanie typu 4.1.1 oproti 4.1.2 je 50%. Dôvod prečo som sa rozhodol len pre 20% šancu na vykonanie druhého variantu je ten, že výber/generovanie suseda je náhodné a susedia nie sú unikátny. Ak by šanca na hľadanie pomocou druhého variantu bola vysoká, tak by sa program teoreticky mohol dostať len pomocou pár iterácií do úplne odlišného stavu a to by znamenalo, že neprebieha lokálne vylepšovanie ale skôr náhodne hľadanie.



Obrázok 13: Implementácia prvého variatnu - a), b)

### Prehodenie pozícii dvoch génov

Pri prvej implementácií algoritmu program hľadal nových susedov len prehodením 2 náhodnych génov.



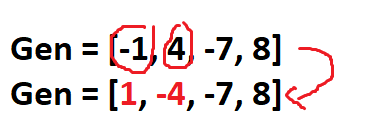
Obrázok 14: Ukážka realizácie prvého variantu - a)

Počet susedov sa pri tejto metóde rovná:

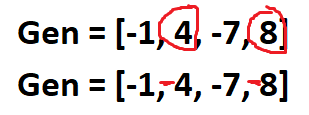
Pre počet génov (n) = 4 je týmto hľadaním program schopný nájsť sedem (3 + 2 + 1) unikátnych susedov. S výsledkami program využitím tejto jedinej metódy hľadania som nebol spokojný a preto som sa rozhodol implementovať variant b) hľadania susedov.

### Inverzia rozhodovania dvoch náhodných génov

Implementácia tohto spôsobu je veľmi podobná 4.1.1. Vyberú sa 2 náhodné gény a zmenia sa ich znamienka na opačné.



Obrázok 15: Prvá ukážka realizácie prvého variantu – b)

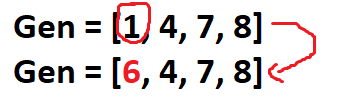


Obrázok 16: Druhá ukážka realizácie prvého variantu – b)

Počet susedov sa po pridaní tohto spôsobu zvýši o:

## Druhý variant

Druhý variant spočíva vo výbere náhodneho génu a jeho nahradení génom, ktorý sa v chromozóme nenachádza. Rozhodujúca je štartovacia pozícia génu v prípade nového génu 29 sa gén nevymení s náhodným ak sa v chromozóme už nachádza gén -29.



Obrázok 17: Ukážka realizácie druhého variantu

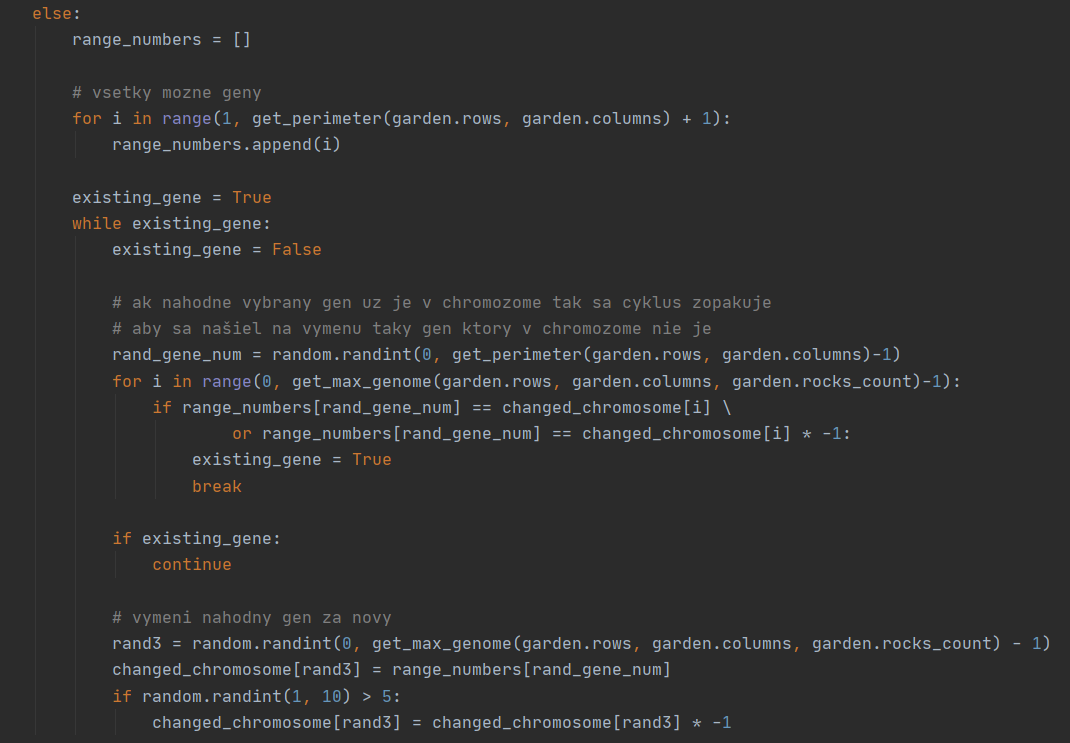
Pridaním druhej varianty hľadania susedov sa ich počet zvýší o:

**x\*n**

o = obvod záhrady

n = počet génov v chromozóme

x = o – n (počet génov ktoré nie sú v chromozóme)



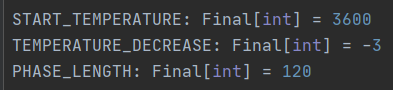
Obrázok 18: Aktuálna implementácia druhé variantu

Druhý variant som najprv implementoval len ako mutáciu, šanca mutácie sa zvyšovala keď sa program zasekol v lokálnom maxime. Po dlhšom uvažovaní som ale program zmenil aby sa toto hľadanie susedov vykonávalo s určitou šancou vždy, pri každom hľadaní susedov. Aktuálne je v programe nastavená šanca na výber suseda touto metódou na 20%.

# Simulované žihanie

Simulované žíhanie je pravdepodobnostná optimalizačná metóda prehľadávania stavového priestoru založená na simulácii žíhania ocele. Pri prehľadavaní stavového priestoru môže ľahko dôjsť ku uviaznutiu v lokálnom maxime. V simulovanom žíhani sa tomu snažíme zabrániť tým, že robíme aj zmeny k horšiemu. Na základe šance si algoritmus vyberie aj suseda, ktorý má horšie ohodnotenie ako aktuálny stav. Tieto zmeny nastávajú hlavne zo začiatku a vďaka nim sa môžeme dostať z lokálneho maxima. Šanca zmeny záleží na teplote. Čím väčšia teplota tým vačšia šanca na vykonanie zmeny. Pri nízkej teplote sa simulované žihanie začne správať ako lačný algoritmus. Simulované žihanie pracuje len s jedným kandidátnym riešním.

V programe je simulované žihanie možné kontrolovať troma konštantami – začiatočnou teplotou, znižovaním teploty a dĺžkou jednej fázy.



Obrázok 19: Príklad nastavenia simulovaného žíhania

Algoritmus simulovaného žihania je implementovaný vo funkcii *simulated\_annealing()*.



Obrázok 20: Implementácia simulovaného žíhania

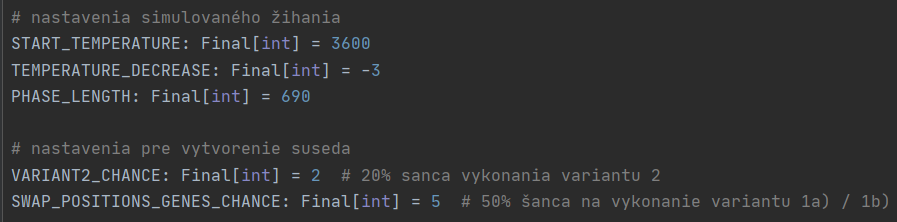
V prípade kedy sa ohodnotenie aktuálneho mnícha rovná ohodnoteniu riešenia tak funkcia vráti riešenie. V prípade kedy sa riešenie nenájde, tak funkcia vráti najlepšieho nájdeného mnícha.

Algoritmus prejde do suseda ak je jeho ohodnotenie lepšie ako ohodnotenie aktuálneho stavu alebo v prípade pravdepodobnostného prijatia nižšieho ako 100% - náhodne číslo v rozmedzí 0 až 1 má nižšiu hodnotu ako .

## Úprava kvôli testovaniu

Kvôli testovaniu som do funkcie pre simulované žíhanie pridal ďalšie premenné a funkcionality na zaznamenávanie priebehu fitness v jednotlivých fázach. Pridané časti sú v zdrojovom kóde okomentované aby bol rozdiel oproti obrázku č. 20 jednoznačný. Program po každom vykonaní algoritmu vygeneruje xls súbor obsahujúci informácie o priebehu fitness v danej teplote. Ak prebieha viac ako 1 test tak sa tento súbor vždy nahrádza a na konci sú uložené len výsledky z posledného vykonania algoritmu.

# Testovanie a výsledky pre rôzne nastavenia



Obrázok 21: Nastavenie 1 pre simulované žíhanie

Pri určovaní dĺžky fázy som sa odrážal hlavne od výpočtu koľko unikátnych susedov môže daný stav v mojom programe mať, v prípade, že sa žiadny so susedov nevybral až dokedy sa neprejdú všetky.

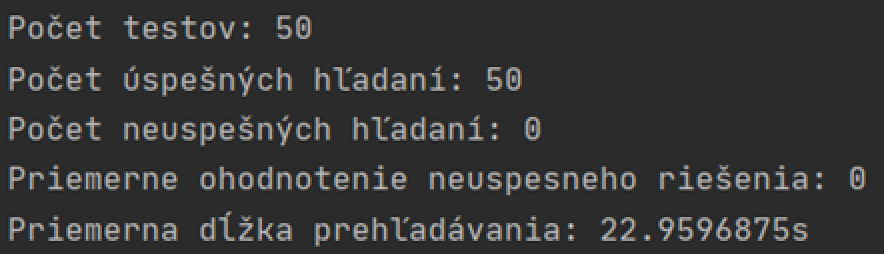
2\*( ) + x\*n = 2(120) + 448 = 688

Ku všetkým hodnotám som sa ale viac menej ajtak dopracoval len formou testovania a hľadaním optimálneho nastavenia. To isté platí aj pre nastavené hodnoty v hľadaní suseda. Pri hľadaní optimálneho nastavenia som sa viac sústredil na čo najvyššiu úspešnosť než na rýchlosť vykonania.Nastavenie z obrázka 21 dosahuje pre záhradu zo zadania 100% úspešnosť ale nie je najrýchlejšie.

## Testovanie – nastavenie 1

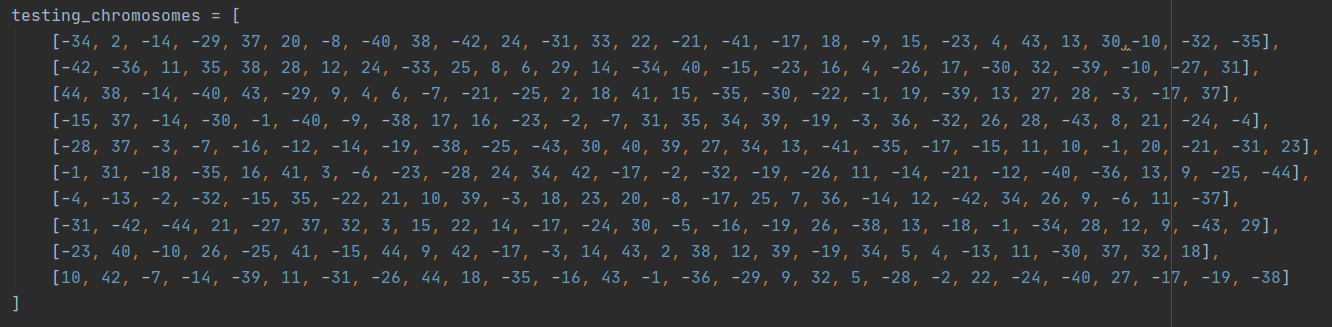
Pre nastavenie z obrázku 21 bolo vykonaných 200+ testov, všetky skončili s úspešným najdením riešenia (100% úspešnosť). Priemerná dĺžka prehľadávania pre toto nastavenie sa pohybuje okolo 20 sekúnd a odvíja sa hlavne od začiatočného chromozómu. Samozrejme je možné považovať toto nastavenie za pomalé a radšej zvoliť nastavenie s nižšou úspešnosťou ale rýchlejším vykonaním. To už ale záleží od toho čo považujeme za optimálne nastavenie.

### Testovanie nastavenia s náhodnými chromozónmi

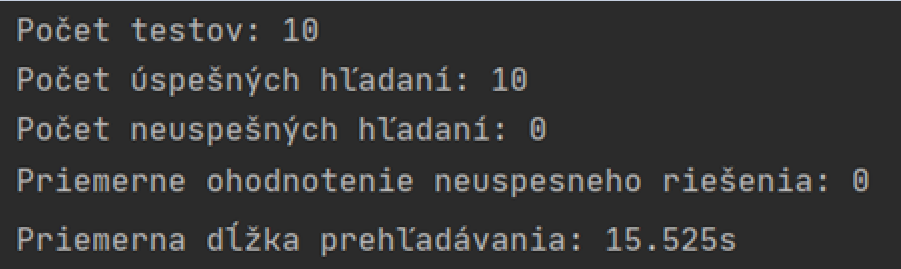


Obrázok 22: Výsledok testovania s náhodnými chromozónmi – nastavenie 1

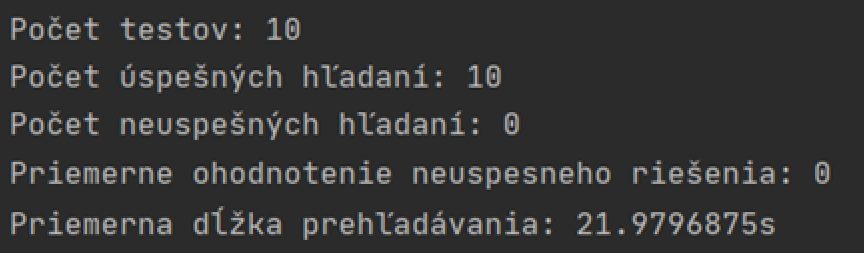
### Testovanie nastavenia s určenými chromozónmi



Obrázok 23: Určené začiatočné/prvé chromozómy

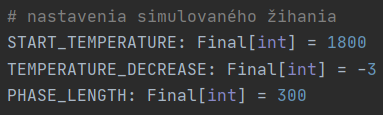


Obrázok 24: Výsledok testovania so zadanými chromozónmi pre nastavenie 1 – pokus č. 1



Obrázok 25: Výsledok testovania so zadanými chromozónmi pre nastavenie 1 - pokus č. 2

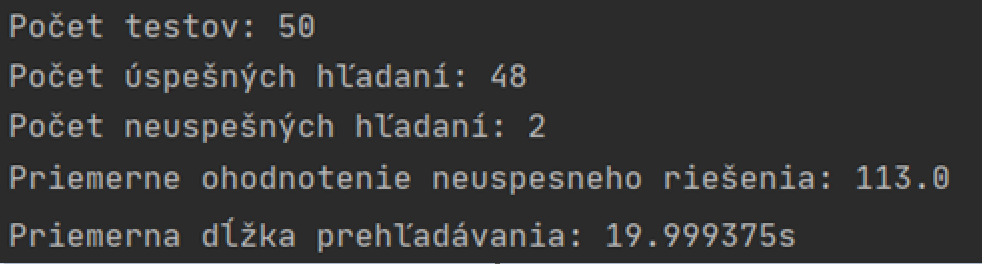
## Testovanie – nastavenie 2



Obrázok 26: nastavenie 2

Pri hľadaní nastavenia s ktorým by som porovnal prvotné nastavenie som strávil veľa času vzhľadom na to, že som sa snažil nájsť rovnováhu medzi najrýchlejším vykonaním a  najvyššou úspešnosťou. Nakoniec som sa dopracoval ku nastaveniu z obrázka 26. Nižšie hodnoty v nastavení buď dosahovali horšie časy vykonania alebo dosahovali horšiu úspešnosť. Toto nastavenie už naďalej nedosahuje 100% úspešnosť vo všetkých prípadoch testovania ale jeho vykonanie je rýchlejšie.

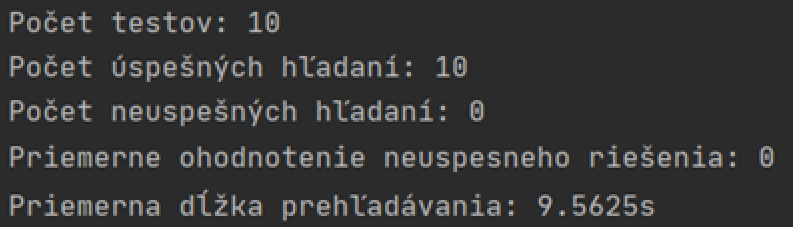
### Testovanie nastavenia s náhodnými chromozónmi



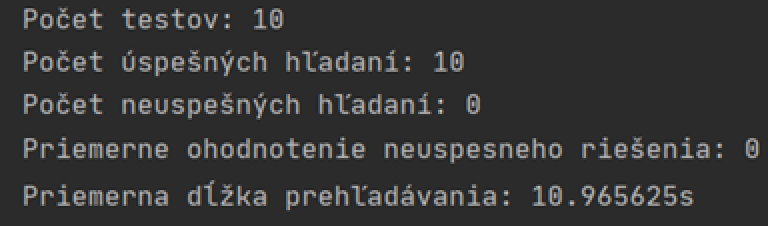
Obrázok 27: Výsledok testovania s náhodnými zač. chromozónmi – nastavenie 2

### Testovanie nastavenia s určenými chromozónmi

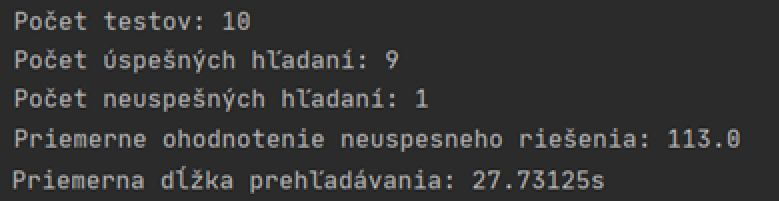
Pri tomto testovaní boli využité začiatočné/prvé chromozómy z obrázka 23.



Obrázok 28: Výsledok testovania so zadanými chromozónmi pre nastavenie 2 – pokus č. 1



Obrázok 29: Výsledok testovania so zadanými chromozónmi pre nastavenie 2 - pokus č. 2



Obrázok 30: Výsledok testovania so zadanými chromozónmi pre nastavenie 2 - pokus č. 3

## Porovnanie nastavení

Nastavenie 2 môžeme považovať za rýchlejšie, hlavne ak sa dokáže dopracovať ku riešeniu. Vzhľadom na to, že v prípade tohto nastavenia je začiatočna teplota a aj dĺžka fázy nižšia tak sa algoritmus rychlejšie stane pažravým. Kvôli tomu je aj vyššia šanca, že sa zasekne v lokálnom maxime a to spôsobí, že celkové vykonanie bude dlhšie ako v prípade nastavenia 1.

### Priebeh fitness

Priebeh fitness pre záhradu zo zadania a tento chromozóm:

[-34, 2, -14, -29, 37, 20, -8, -40, 38, -42, 24, -31, 33, 22, -21, -41, -17, 18, -9, 15, -23, 4, 43, 13, 30,-10, -32, -35].

V dolnej časti grafu sa nachádza aktuálna teplota. V rámci tohto testovania sa nastavenie 1 dostalo k riešeniu rýchlejšie. Na grafu si môžeme všimnúť, že čím je teplota nižšia tým menej dochádza ku kolísaniu priemernej fitness.

# Záver

Úspešne som vymyslel a implementoval riešenie pre problém zen záhrady využitím algoritmu simulovaného žíhania.

## Dolaďovanie a zlepšovanie

Vo všetkých kapitolach dokumentácie som sa snažil dávať dôraz na to ako som algoritmus a jeho nastavenia pri budovaní dolaďoval a vylepšoval.