SLOVENSKÁ TECHNICKÁ UNIVERZITA

Fakulta informatiky a informačných technológií

v Bratislave

Umelá inteligencia

Zadanie 3

Adam Tomčala

Prednášajúci: Ing. Lukáš Kohútka, PhD.

Cvičiaci: Ing. Boris Slíž

Čas cvičení: Streda 15:00

Adam Tomčala
ID: 103164

Definovanie problému:

Mojou úlohou je upraviť Zenovú záhradku, ktorá môže mať ľubovoľný rozmer **MxN**. Záhradka môže obsahovať políčka, ktoré sa upraviť nedajú (kamene/prekážky). Mních, ktorý záhradku upravuje sa môže pohybovať len zvislo alebo vodorovne, nikdy nie šikmo. Vznikajú nasledujúce pásy:

Políčka, ktoré mních už upravil sa tiež berú ako prekážky. Ak mních narazí na prekážku nastávajú nasledujúce situácie:

- ak mních narazí na prekážku a zároveň sa môže otočiť ľubovoľným smerom (vpravo aj vľavo) je na ňom, ktorý smer si vyberie
- ak mních narazí na prekážku a môže sa otočiť len jedným smerom, vyberie si ten, ktorý je dostupný
- ak mních narazí na prekážku a nemá sa kam otočiť, mních sa zasekne, koniec.

Mních si môže vybrať, ktorým vstupom do záhradky vstúpi (ak sa na vstupnom políčku nenachádza prekážka).

Úprava záhradky končí vtedy, ak mních upraví všetky políčka v záhradke (úspešné ukončenie) alebo vtedy, keď sa mních nebude môcť otočiť (zasekne sa).

Pohyby mnícha:

- HORE
- DOLE
- VĽAVO
- VPRAVO

Adam Tomčala
ID: 103164

Tabu search:

Popis algoritmu

- Tabu vyhľadávanie je metaheuristická vyhľadávacia metóda, ktorá rieši problém zacyklenia v lokálnom extréme.
- Je to v podstate obdoba horolezeckého algoritmu, do ktorého je zavedená krátkodobá pamäť (tabu list), ktorá slúži na zapamätávanie si posledných navštívených stavov.
- Je potrebné si vytvárať od nejakého (náhodne vygenerovaného) stavu nových susedov (stavy, ktoré sú v určitej miere podobné stavu, od ktorého generujeme susedov).
- Po vytvorení susedov hľadáme najlepšieho z nich, porovnávame ho s celkovým doteraz nájdeným najlepším stavom.
- Následne tento stav vložíme do tabu listu.
- Dôležitým faktorom pri tomto algoritme je práve veľkosť tabu listu, pretože jeho veľkosť môže ovplyvniť jeho efektivitu.

- Moje zadanie som programoval v jazyku Python 3.9.

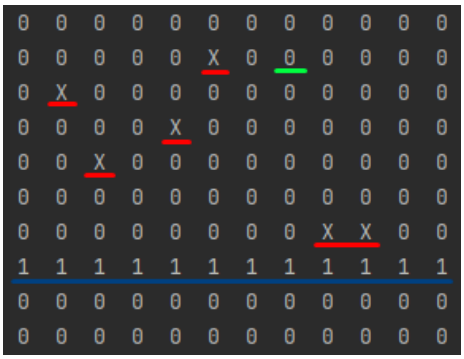
- Použité importy:

```
import random
import time
```

Adam Tomčala
ID: 103164

Reprezentácia záhradky:

- Záhradku, ktorú musí mních upraviť si načítavam zo vstupného textového súboru „input.txt“.
- V programe so záhradkou pracujem ako s 2D listom.
- Políčka, ktoré mních musí upraviť označujem hodnotou **0**.
- Políčka, ktoré predstavujú prekážku označujem hodnotou **X**.
- Ťahy mnícha označujem aktuálnym číslom ťahu.
- Vizualizácia záhradky:



- Prekážky
- Neupravené políčka
- Aktuálny ťah mnícha

- V mojom programe na to slúži funkcia *nacitaj_zahradu()* :

```
def nacitaj_zahradu(vstupny_subor):
    """
    Funkcia nacita zahradu zo suboru ulozi ju do 2D listu.
    :param vstupny_subor:   Subor, z ktoreho citam.
    :return:                Vratim zahradu v 2D liste.
    """
    z = []
    for riadok in vstupny_subor:
        z.append(list(riadok[:len(riadok) - 1]))
    return z
```

- Následne pre vstupnú záhradku zistím jej fitness hodnotu. V tomto prípade fitness znázorňuje celkový počet neupravených políčok v danej záhradke (na začiatku).
- V mojom programe mi na to slúži nasledujúca funkcia *max_fitness_zahrady()* :

```
def max_fitness_zahrady(z):
    """
    Funkcia vrati pocet upravitelnych policok v zahrade.
    :param z:   Vstupna zahrada.
    :return:    Fitness zahrady. -> maximalny pocet policok, ktory sa da upravit.
    """
    fitness = 0
    for riadok in z:
        for policka in riadok:
            if policka == '0':
                fitness += 1
    return fitness
```

Adam Tomčala
ID: 103164

Reprezentácia mnícha:

- Mnícha, ktorý upravuje zadanú záhradku reprezentujem ako list, ktorý obsahuje nasledujúce hodnoty:
- [fitness aktuálneho mnícha , rozhodnutia mnícha , vstupné políčka do záhradky]
- Funkcia, ktorá slúži na vytvorenie mnícha je *vytvor_mnicha()* :

```
def vytvor_mnicha(z):
    """
    Funkcia vytvara mnicha.
    ->  fitness mnicha na zaciatku. 0
    ->  vstupy mnicha do zahrady.   list[0, .. , MxN zahrady]
    ->  rozhodnutia mnicha.         list[1, -1, -1, 1, 1, ...]

    :param z:   Vstupna zahrada.
    :return:    Mnich.       -->    list[fitness, rozhodnutia, vstupy do zahrady]
    """

    vstupy = vytvor_vstupy_mnicha(len(z[0]), len(z))        # vytvorenie vstupov do zahrady a nahodne preusporiadanie
    vstupy = random_vstupy(vstupy)

    rozhodnutia = vytvor_rozhodnutia(z)                     # vytvorenie rozhodnuti mnicha

    return [0, rozhodnutia, vstupy]
```

- Fitness každého mnícha je na začiatku **0**, pretože mních ešte neupravil žiadne políčko.
- Rozhodnutia mnícha vytváram pomocou funkcie *vytvor_rozhodnutia()* :

```
def vytvor_rozhodnutia(z):
    """
    Funkcia vytvara list rozhodnuti mnicha, vytvorim tolko rozhodnuti, kolko je prekazok v zahrade na zaciatku.
    :param z:   Vstupna zahrada.
    :return:    List rozhodnuti mnicha.
    """
    rozhodnutia = []
    for riadok in z:
        for policka in riadok:
            if policka == 'X':
                rozhodnutia.append(vytvor_rozhodnutie())

    return rozhodnutia
```

- Rozhodnutia mnícha je list, v ktorom sa nachádzajú hodnoty **1** alebo **-1**. Veľkosť listu „rozhodnutia“ je určená počtom prekážok v začiatočnej záhradke.
- Prechádzam začiatočnú záhradku, ak narazím na prekážku, zavolá sa funkcia *vytvor_rozhodnutie(),* ktorá vygeneruje náhodný bit.

- Funkcia, ktorá vytvára jednotlivé rozhodnutia:

```
def vytvor_rozhodnutie():
    """
    Funkcia vytvori 1 rozhodnutie mnicha.
    :return:    1 bit, ktory symbolizuje pohyb mnicha (jeho rozhodnutie).
    """
    bit = random.getrandbits(1)
    if bit:
        return bit
    return -1
```

- Vstupné políčka do záhradky vytváram pomocou funkcie *vytvor_vstupy_mnicha()* :

```
def vytvor_vstupy_mnicha(sirka, vyska):
    """
    Funkcia vytvori list, ktory obsahuje vstupy do zahrady. Pocita sa to na zaklade obvodu zahrady.
    :param sirka:   Sirka zahrady.
    :param vyska:   Vyska zahrady.
    :return:        Vstupy do zahrady.
    """
    return [x for x in range(2*(sirka+vyska))]
```

- Vstupné políčka mnícha sú reprezentované v liste, v podstate sú to obvodové políčka záhradky.
- Vstupné políčka ešte náhodne poprehadzujem vo funkcii *random_vstupy()*.
- Vstupné políčka prehadzujem náhodne preto, aby mních vstupoval do záhradky náhodne.

Pohyb mnícha po záhradke:

- Najskôr sa vygenerujú vstupné pozície mnícha ako bolo uvedené vyššie.
- Vstupné políčka mnícha do záhradky je list obvodových políčok záhradky, ktoré sú náhodne poprehadzované.
- List pre záhradku 10x12 bude vyzerať nasledovne -> [0, 1, ... , obvod – 1 / 43]
- Vizualizácia:

	0	1	2	3	4	5	6	7	8	9	10	11	
43	0	0	0	0	0	0	0	0	0	0	0	0	12
42	0	0	0	0	0	X	0	0	0	0	0	0	13
41	0	X	0	0	0	0	0	0	0	0	0	0	14
40	0	0	0	0	X	0	0	0	0	0	0	0	15
39	0	0	X	0	0	0	0	0	0	0	0	0	16
38	0	0	0	0	0	0	0	0	0	0	0	0	17
37	0	0	0	0	0	0	0	0	X	X	0	0	18
36	0	0	0	0	0	0	0	0	0	0	0	0	19
35	0	0	0	0	0	0	0	0	0	0	0	0	20
34	0	0	0	0	0	0	0	0	0	0	0	0	21
	33	32	31	30	29	28	27	26	25	24	23	22	

- Čísla v červených rámoch sú vstupné pozície mnícha.
- Čísla v **čiernom** štvorci predstavujú vstupnú záhradku.

- Vyberie sa prvé políčko z tohto listu (napr. políčko 6).
- Potom ako sa vyberie políčko, zistím si súradnicu daného políčka v záhradke pomocou funkcie *najdi_vstupnu_poziciu()* :

```python
def najdi_vstupnu_poziciu(vstup, sirka, vyska):
    """
    Funkcia urci suradnice vstupneho policka mnicha na zaklade sirky a vysky zahrady.

       0 1 2
     9       3       --> Vstupy do zahrady
     8       4
       7 6 5

    Vyberiem napr. vstup 1 -> jeho suradnice:   [0,1]

    :param vstup:   Konkretny vstup do zahradky.    Int
    :param sirka:   Sirka zahrady.                  Int
    :param vyska:   Vyska zahrady.                  Int
    :return:        Suradnice vstupneho policka.    [riadok - 1, stlpec - 1]
    """
    if vstup < sirka:
        return [0, vstup % sirka]
    elif vstup < sirka + vyska:
        return [abs(vstup - sirka) % vyska, sirka - 1]
    elif vstup < 2*sirka + vyska:
        return [vyska - 1, sirka - (abs(vstup - sirka - vyska) % sirka) - 1]
    else:
        return [vyska - (abs(vstup - 2*sirka - vyska) % vyska) - 1, 0]
```

- Funkcia na základe toho, na ktorej strane obvodu záhradky sa daný vstup nachádza vypočíta súradnicu aktuálneho vstupného políčka.
- Pre vstupné políčko **6** bude vstupná pozícia **[0, 6]**.
- Ďalej sa na základe vstupného políčka musí určiť smer, akým sa mních bude pohybovať.
- Na to slúži funkcia *urci_smer()* :

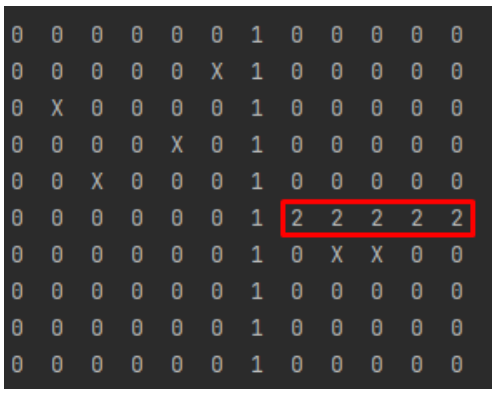
```python
def urci_smer(vstup, sirka, vyska):
    """
    Funkcia urci smer mnicha na zaklade vstupu do zahradky.
    :param vstup:   Vstup do zahradky.  Int
    :param sirka:   Sirka zahrady.      Int
    :param vyska:   Vyska zahrady.      Int
    :return:        Smer mnicha         Char
    """
    if vstup < sirka:
        return 'D'
    elif vstup < sirka + vyska:
        return 'L'
    elif vstup < 2*sirka + vyska:
        return 'U'
    elif vstup < 2*sirka + 2*vyska:
        return 'R'
```

- Funkcia určí smer mnícha na základe toho, odkiaľ mních do záhradky vstupuje.
- Pre vstupné políčko **6** bude smer mnícha **D – dole**.
- Takto bude vyzerať záhradka, ak bude vstupné políčko **6**.

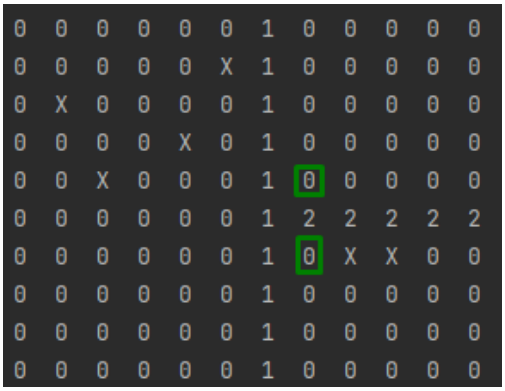
```
0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 X 1 0 0 0 0 0
0 X 0 0 0 0 1 0 0 0 0 0
0 0 0 0 X 0 1 0 0 0 0 0
0 0 X 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 X X 0 0
0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0
```

- Toto bol zobrazený / vysvetlený pohyb, ak sa v ťahu nenachádza prekážka.

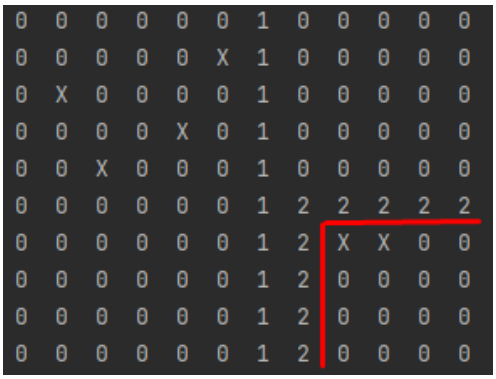
- Ak by bol ďalší vstup **17,** mních narazí na prekážku (políčko, ktoré už upravil):



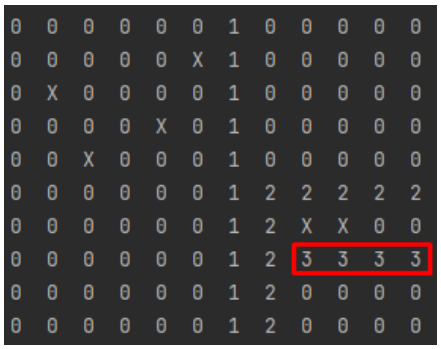
- Idú sa kontrolovať políčka, do ktorých by sa mních mohol pohnúť.
- Skontroluje nasledovné políčka:



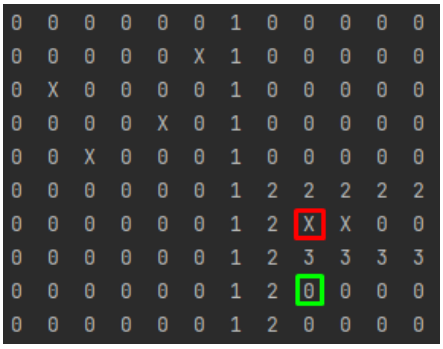
- V tomto prípade nie je upravené ani jedno z políčok, to znamená, že mních sa môže rozhodnúť na základe svojich rozhodnutí.
- Mních sa rozhodol nasledovne a výsledný pohyb vyzerá takto:



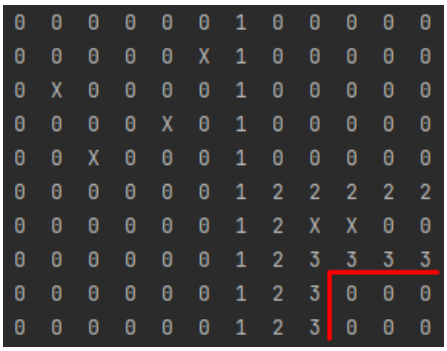
- Ak by bolo ďalšie vstupné políčko **19**, mních opäť narazí na prekážku:



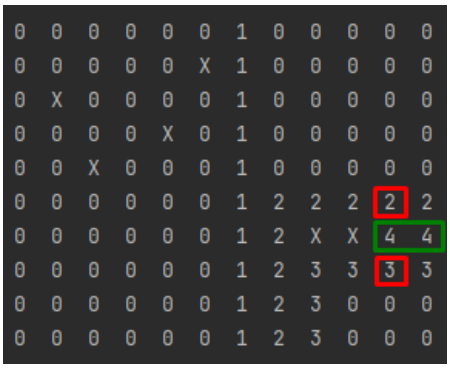
- Opäť sa kontrolujú obe políčka, do ktorých sa mních môže posunúť:



- Z obrázku vyplýva, že jedno z políčok je prekážka, tým pádom mních si musí vybrať to políčko, do ktorého sa môže pohnúť.
- Pohyb sa dokončí takto:



- Ak by bolo ďalšie vstupné políčko s číslo **18** pohyb bude vyzerať takto:



- Mních narazí na prekážku, opäť sa skontrolujú obe políčka, do ktorých sa mních môže posunúť. Obe sú prekážky a mních sa zasekol (koniec úpravy záhradky).

Adam Tomčala
ID: 103164

Vytváranie nových mníchov:

- Nových mníchov vytváram vo funkcií *vytvor_susedov()*:

```
def vytvor_susedov(m, vst_subor):
    """..."""
    s = []
    for index in range(10):
        novy_sused = [0, [y for y in m[1]], [z for z in m[2]]]
        novy_sused[1][index % len(m[1])] *= -1
        for j in range(10):
            r_num1 = random.randint(0, len(m[2]) - 1)
            r_num2 = random.randint(0, len(m[2]) - 1)
            if r_num1 == r_num2:
                j -= 1
                continue
            novy_sused[2][r_num1], novy_sused[2][r_num2] = novy_sused[2][r_num2], novy_sused[2][r_num1]

        s.append(novy_sused)

    for novy_sused in s:
        z = nacitaj_zahradu(vst_subor)
        vst_subor.seek(0)
        uprav_zahradu(z, novy_sused, False)
        novy_sused[0] = fitness_mnicha(z)

    return s
```

- Funkcia vytvára 10 nových mníchov, ktorí sa vytvárajú nasledovne:
    - Vytvorím nového mnícha, ktorý bude rovnaký ako mních, podľa ktorého vytváram nových
    - Každému novovzniknutému mníchovi zmením jedno rozhodnutie na opačné
    - Každému novovzniknutému mníchovi náhodne zmením 10 vstupov do záhradky
- Následne s každým susedom upravím začiatočnú záhradku, a tak zistím jeho fitness hodnotu.

Tabu vyhľadávanie:

- Ako prvé si vytvorím náhodného mnícha, s ktorým upravím záhradku a zistím jeho fitness hodnotu.
- Následne iterujem v cykle, až kým nenájdem požadovanú hodnotu.
- Zoradím si susedov od najväčšej fitness hodnoty.
- Zistím, čí sa daný mních (sused) nachádza v tabu list.
- Ak sa nachádza, vymažem ho z listu susedov. Ak sa nenachádza, ponechám ho v liste.
- Najlepší kandidát bude najlepší mních, ktorý ostal v liste susedov.
- Porovnám ho či je lepší ako doteraz najlepšie nájdené riešenie.
- Ak áno, prepíšem najlepšie doteraz nájdené riešenie.
- Vložím najlepšieho kandidáta do tabu listu.
- Ak som našiel požadované riešenie alebo som prekročil najviac dovolený počet iterácií, ukončím cyklus. Inak pokračujem v hľadaní.

- Kód tabu vyhľadávanie v mojom programe:

```
while najlepsi[0] != max_fitness:                    # Cyklus iteruje, kym sa nenajde vysledok
    susedia = vytvor_susedov(najlepsi_kanditat, subor)   # Vytvorenie susedov
    susedia.sort(key=lambda x: x[0], reverse=True)       # Usporiadanie podla fitness

    for sused in susedia:                            # Odstranenie susedov
        if sused in tabu_list:                       # ktori sa nachadzaju v tabu liste
            susedia.remove(sused)

    najlepsi_kanditat = susedia[0]                   # najlepsi kanditat je najlepsi
                                                     # z danych susedov
    # print(susedia[0][0])

    if najlepsi_kanditat[0] > najlepsi[0]:           # Ak je novonajdeny kandidat lepsi
        najlepsi = najlepsi_kanditat                 # ako celkovo najlepsi, prepisem ho

    najlepsii.append(najlepsi_kanditat[0])
    tabu_list.append(najlepsi_kanditat)              # Pridam najlepsieho suseda do tabu listu

    if len(tabu_list) > tabu_list_size:              # Ak sa v tabu liste nachadza viac mnichov
        vymaz_prveho(tabu_list)                      # ako je dovolene odstranim prveho

    i += 1
    if i > 5000:                                     # Ak cyklus prekroci 5000 iteracii,
        break                                        # zastavim hladanie
```

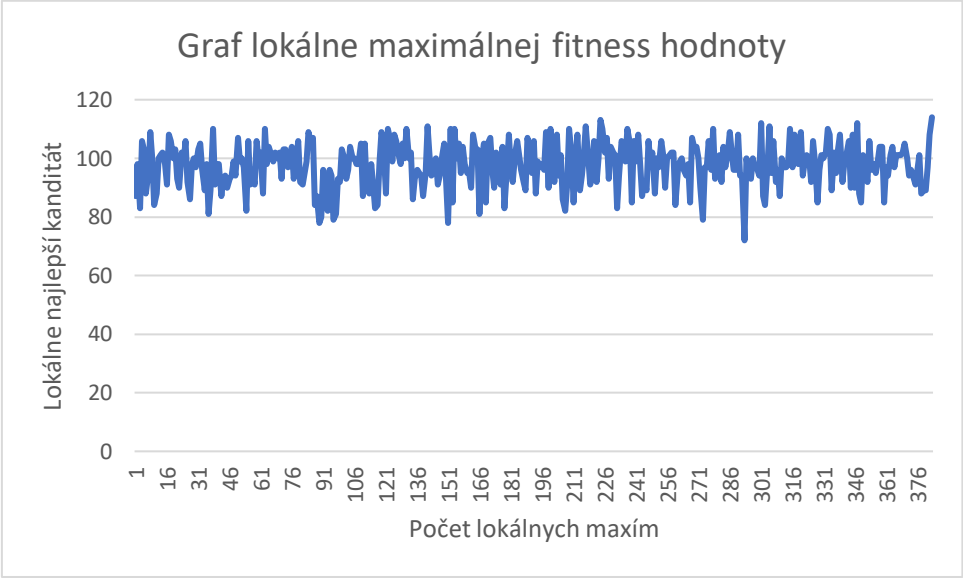
Testovacie scenáre:

- V mojom riešení som testoval 2 vstupné záhradky, obe majú rôzne rozmery:
  - Mapa číslo 1 - 12x10, ľahšia obtiažnosť (mapa zo zadania)
  - Mapa číslo 2 - 8x8, ťažšia obtiažná mapa (malá mapa s relatívne väčším počtom prekážok)
- Pri všetkých vstupoch som menil aj maximálne veľkosti samotného tabu list (použil som veľkosti 10, 25, 50).
- V testovanie som uskutočňoval tak, že som danú záhradku spustil 100 krát a meral som nasledovné parametre:
  - Ako sa počas riešenia menila maximálna lokálna fitness hodnota
  - Priemerný čas vyriešenia danej záhradky pri rôznych veľkostiach tabu listu
  - Priemerný počet iterácií

Adam Tomčala
ID: 103164

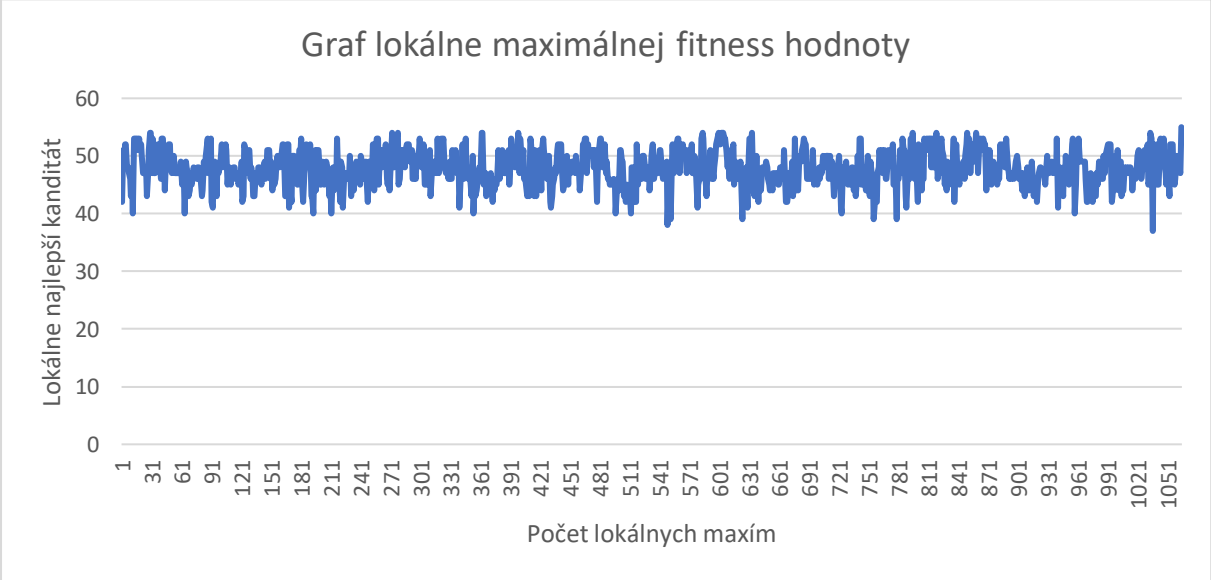
Grafy lokálne maximálnych fitness hodnôt:

- **Mapa číslo 1:**



  - Maximálna fitness hodnota záhrady = 114

- **Mapa číslo 2:**



  - Maximálna fitness hodnota záhrady = 55

Priemerné časy riešení pri rôznych veľkostiach tabu listu (počet vykonaní: 100):

Veľkosť tabu listu	Mapa číslo 1	Mapa číslo 2
10	0,1145 s	0,3458 s
25	0,1345 s	0,3380 s
50	0,1554 s	0,3886 s

Celkové časy riešení pri rôznych veľkostiach tabu listu:

Veľkosť tabu listu	Mapa číslo 1	Mapa číslo 2
10	11,4571 s	34,5855 s
25	13,4507 s	33,8025 s
50	15,5489 s	38,8666 s

Priemerný počet iterácií:

	Mapa číslo 1	Mapa číslo 2
Priemerný počet iterácií	67	307

Zhodnotenie testovania:

- Testovaním som dokázal, že tabu algoritmus sa dokáže dostať z lokálnych maxím tým, že vchádza aj do horších stavov.
- Taktiež som potvrdil, že veľkosť tabu listu ovplyvňuje efektivitu algoritmu.