Slovenská technická univerzita

Fakulta informatiky a informačných technológií

 Ilkovičova 3, 842 19 Bratislava

**Ema Richnáková**

**Evolučný algoritmus**

**Zenová záhrada**

Predmet: Umelá inteligencia

Akademický rok: 2019/2020

1. *Úvod*

 Evolučný algoritmus v zenovej záhrade používa metódu kríženia two-point, čiže rozdel

1. *Opis riešenia*
   1. *Implementácia*

Zdrojový kód programu je napísaný v programovacom jazyku Python.

* 1. *Reprezentácia údajov v kóde*

Údaje v kóde, ktoré ovplyvňujú veľkosť riešeného problému:

* grid\_size = (x, y)
  + nachádza sa v bloku main
  + x predstavuje šírku šachovnice, y predstavuje výšku šachovnice
* num\_of\_starting\_position = x
  + nachádza sa ako globálna premenná na začiatku kódu
  + určuje koľkokrát sa Eulerov kôň spustí z rôznych štartovacích pozícii, ktoré sú generované náhodne
  1. *Použitý algoritmus*

Využitý je algoritmus Eulerov kôň, alebo pod iným meno ako Jazdcova prechádzka. Výber vhodného nasledujúceho ťahu je riešené heuristicky.

Pravidlá na úspešné dokončenie Eulerovho koňa:

1. Každý ťah je vykonaný podľa šachových pravidiel.
2. Každé pole šachovnice je navštívené práve raz.

Heuristika sa riadi podľa Warnsdorffovo pravidla. To znie nasledovne:

V tomto algoritme je za pole, na ktoré jazdec potiahne, vybrané vždy to, z ktorého je možné ďalej pokračovať najmenej spôsobmi. Vďaka tomu sú častejšie obsádzané polia, ktoré sú vlastne ,,skoro nedostupné“ a naopak zatiaľ ľahko dostupné polia sa nechávajú na neskôr.

Problém sa dá riešiť dvoma spôsobmi, buď je riešenie tzv. ,,uzavreté‘‘ alebo ,,otvorené‘‘.

Uzavreté riešenie je také riešenie, kedy po úspešnom prejdení šachovnice jazdec z koncového poľa sa vie jedným ťahom dostať späť na pole, z ktorého vychádzal a tým ,,uzavrie“ cestu. Pri otvorenom riešení nie je takáto možnosť.

V tomto riešení je použité otvorené riešenie.

([zdroj](https://cs.wikipedia.org/wiki/Jezdcova_proch%C3%A1zka#Warnsdorff%C5%AFv_algoritmus))

* 1. *Riešenie*

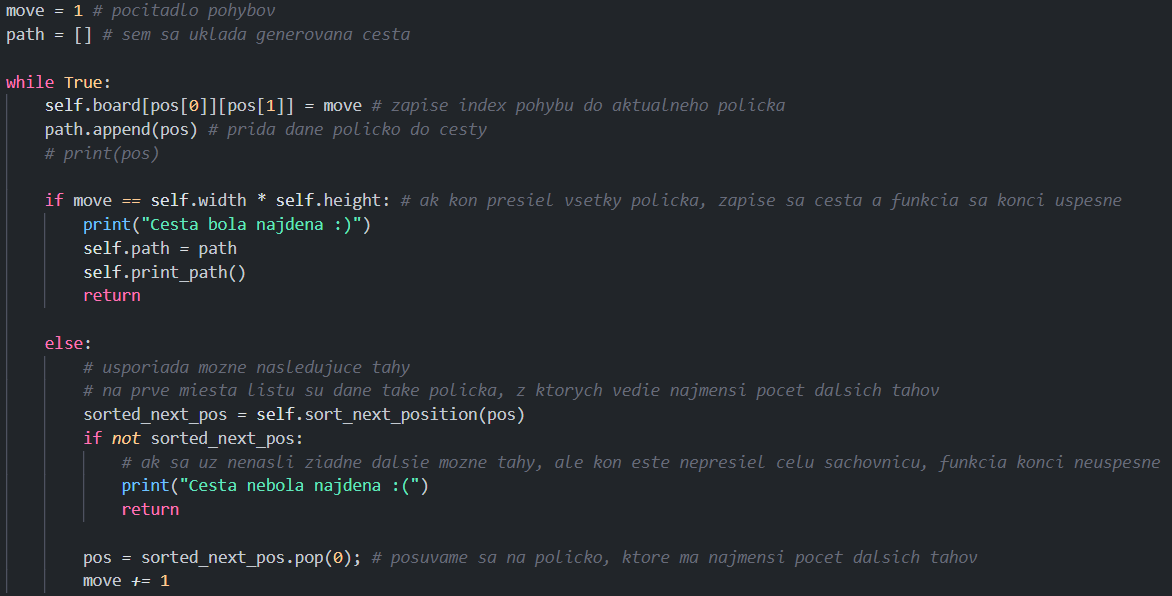
Na začiatku sa vygeneruje šachovnica o veľkosti, ktorú zadá užívateľ. Ak nezadá veľkosť a stlačí klávesu *Enter*, automaticky šachovnica bude o veľkosti 8x8. Šachovnica ako pole má začiatočný bod (0, 0) v ľavom hornom rohu.

Počet spustení Eulerovho koňa je podľa zadania nastavené na 10. Toto nastavenie sa dá zmeniť zásahom do zdrojového kódu, konkrétne prepísaním globálnej *premennej num\_of\_starting\_position*.

Pred každým začatím cesty sa náhodne vygeneruje x-ová a y-ová súradnica začiatočného poľa. Generovanie je uskutočnené pomocou knižnice *random*.

Potom sa vytvorí šachovnica, ktorej každé políčko je naplnené číslom 0, čo predstavuje nenavštívené políčka. Tieto políčka sa počas behu programu budú meniť. Bude sa do nich zapisovať poradie pohybu koňa.

Začína sa hľadanie cesty.



  Do funkcie *find\_path()* sa zadá začiatočná pozícia koňa na šachovnici.

Na danú pozíciu zaznačíme číslo, teda ako poradie navštívených políčok.

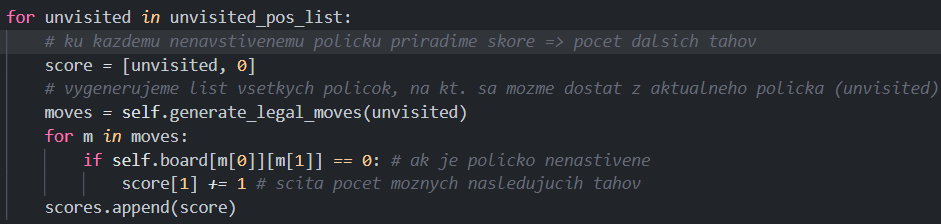
Pridáme políčko do cesty.

Ak sme spravili taký počet krokov ako je počet políčok, funkcia úspešne končí a vypíše cestu koňa. Ak sme ešte nespravili taký počet krokov, aký je počet políčok na šachovnici, môžeme začať hľadať vhodné ďalšie ťahy funkciou *sort\_next\_position(pos)*.

V danej funkcii si pomocou funckie *generate\_legal\_moves(cur\_pos)* vygenerujeme všetky legálne ťahy, ktoré sú možné spraviť zo zadaného poľa *cur\_pos* v rámci šachovnice.

Potom zistíme, pomocou ktorých daných legálnych ťahov sa dostaneme na voľné polia a tie si zapíšeme do listu.

Ak neboli nájdené žiadne ďalšie voľné legálne polia, funkcia vracia prázdny list.



Ak boli nájdené, tieto polia ohodnotí. Hodnotenie poľa (alebo skóre) nadobúda hodnotu počtu ďalších možných voľných ťahov z daného poľa.

Ohodnotené polia sa usporiadajú vzostupne a teda na prvých miestach budú polia s najmenším ohodnotením. Takto usporiadanie polia sa z funkcie *sort\_next\_position(pos)* vrátia, ale bez informácie ohodnotenia.

Ak sa z funkcie *sort\_next\_position(pos*) vrátil prázdny list, znamená to, že sa nenašli žiadne voľné legálne polia a keď Eulerov kôň ešte neprešiel všetky polia, funkcia definitívne končí neúspešne, tzn. nebola nájdená taká cesta, ktorú by vedel kôň prejsť bez toho, aby na šachovnici navštívil políčka viackrát.

A ak sa z funkcie *sort\_next\_position(pos)* vrátil naplnený list, vyberie sa prvý element listu, teda pole s najmenším ohodnotením a pokračuje sa celý tento cyklus z tohto novozískaného poľa.

Nakoniec sa vypíše šachovnica, kde je zaznačená cesta Eulerovho koňa pre vizualizáciu a kontrolu správnosti riešenia.

1. *Testovanie*

Kontrola správnosti riešenia bola overovaná vizuálne pomocou výpisov. Vypisovala som si, aké hodnotenie určovalo pre jednotlivé políčka, vizuálne som to overila a skontrolovala, taktiež výpisom, či bolo hodnotenie správne zoradené, a teda či sa vyberalo pole s najmenším ohodnotením.

Vždy na konci programu si vypisujem šachovnicu s cestou Eulerovho koňa. Ak bolo riešenie úspešne dokončené, vypísala som aj poradie jednotlivých políčok, táto informácia mi dopomáhala ku kontrole cesty na šachovnici.

1. *Zhodnotenie riešenia*

Hľadanie riešenia v probléme Eulerovho koňa pomocou heuristiky (Warnsdorffovo pravidla) a bez vracania sa na predošle prejdené polia je veľmi rýchle a na malých šachovniciach (šachovnice do veľkosti 7x7) aj veľmi úspešné riešenie problému. Pri šachovniciach nad 8x8 táto úspešnosť klesá aj z toho dôvodu, že nie je zaručené, že kôň pôjde najefektívnejšou cestou, lebo vidí len 2 kroky dopredu. Z takejto informácie sa nedá moc predpokladať, ktorá časť možnej cesty bude mať aj úspešné ukončenie.