Eulerov kôň

Zadanie 2 pre predmet Umelá Inteligencia

Matúš Tundér

Zadanie

Úlohou je prejsť šachovnicu legálnymi ťahmi šachového koňa tak, aby každé políčko šachovnice bolo prejdené (navštívené) práve raz. Riešenie treba navrhnúť tak, aby bolo možné problém riešiť pre štvorcové šachovnice rôznych veľkostí (minimálne od veľkosti 5 x 5 do 20 x 20) a aby cestu po šachovnici bolo možné začať na ľubovoľnom východziom políčku.

Úloha (g)

... Implementujte túto heuristiku do algoritmu prehľadávania stromu do hĺbky a pre šachovnicu 8x8 nájdite pre 10 rôznych východzích bodov jedno (prvé) správne riešenie (pre každý východzí bod). Algoritmus s heuristikou treba navrhnúť a implementovať tak, aby bol spustiteľný aj pre šachovnice iných rozmerov než 8x8. ...

Riešenie

Stav

Stav reprezentuje šachovnicu, kde je určené na každom políčku či už bolo navštívené alebo nie pomocou 2D (*bool*) poľa. Zároveň si pamätá veľkosť šachovnice (šírka a výškam, *integer*), aktuálnu pozíciu koňa (X a Y súradnice, *integer*) a predchádzajúci stav (referencia).

Operátory

Každý typ pohybu, ktorý môže kôň spraviť na šachovnici je reprezentovaný operátorom (metódou), čiže dostupný počet operátorov je 8. To či sa dá operátor použiť vieme zistiť na základe aktuálnej pozície koňa, veľkosti šachovnice a už použitých políčok. Cena všetkých operátorov je 1.

Heuristická funkcia

Na vyhľadávanie cesty bolo použité Warnsdorfové pravidlo (H. C. von Warnsdorf, 1823). Princíp tejto heuristiky spočíva v tom, že cena stavu je rovná počtu dostupných operátorov (obr.č.1).

Uzol

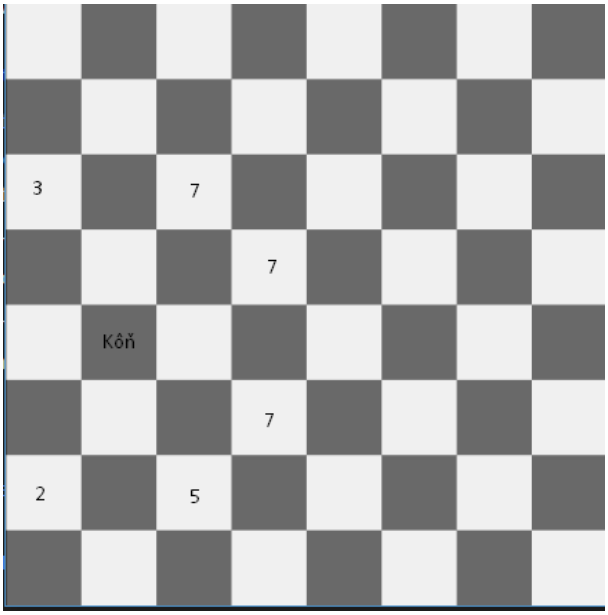
Uzol si okrem aktuálne stavu pamätá počet políčok ktoré ešte treba navštíviť (integer) a použiteľné operátory (*List* obsahujúci dvojice súradníc, na ktoré môže kôň prejsť), kde ich počet zároveň slúži ako cena.

Algoritmus

1. Vytvor počiatočný uzol a vložím ho do zásobníka
2. Ak je nepárna veľkosť šachovnice a začína sa na párnom políčku, skonči – riešenie neexistuje
3. Ak je zásobník prázdny, skonči – riešenie neexistuje
4. Vyber uzol zo zásobníka
5. Ak sa jedná o konečný uzol (počet políčok, ktoré ešte treba navštíviť je 0) skonči a vráť cestu
6. Ak nemôžeš použiť žiaden operátor, choď na krok 3 – slepá ulička
7. Vygeneruj uzly(listy) pre každý dostupný operátor
8. Vlož vygenerované uzly od najdrahšieho po najlacnejší do zásobníka

Matúš Tundér (74288)
Utorok 8:00

9. Choď na krok 3



obr.č.1 – Cena nasledujúcich stavov

Implementácia

Pre vyriešenie tejto úlohy som si zvolil jazyk C# 6 na .Net Framework-u 4.5.2. Príklad aplikácie je možné vidieť na obrázku č.2. Hlavná logika je osamostatnená v Dll, v ktorej sa nachádza upravený algoritmus spomenutý vyššie. Hlavnou zmenou je pridanie časovača, ktorý má na starosti aby sa neprehľadávalo dlhšie než povolený čas. Aplikácia si taktiež pamätá počet vygenerovaných/navštívených stavov a celkový čas potrebný na vyhľadávanie.

Knižnica obsahuje statickú triedu Operatory, v ktorej sa nachádzajú metódy, ktoré ak je to možné vracajú dostupné pozície pre aktuálny stav, inak vrátia *null*. Internal triedu Uzol v ktorej sa nachádza referencia na triedu Stav. Obe tieto triedy sú obohatené o metódy, ktoré dokážu vytvoriť novú inštanciu na základe predošlého stavu/uzlu. K uzlu po vytvorení ešte dokážeme priradiť dostupné operátory. Nakoniec je tam verejná trieda Search, ktorá má v sebe vyhľadávací algoritmus a atribúty určené pre štatistiku. Podrobnejší popis je možné pozrieť v komentároch a Summary zdrojového kódu.

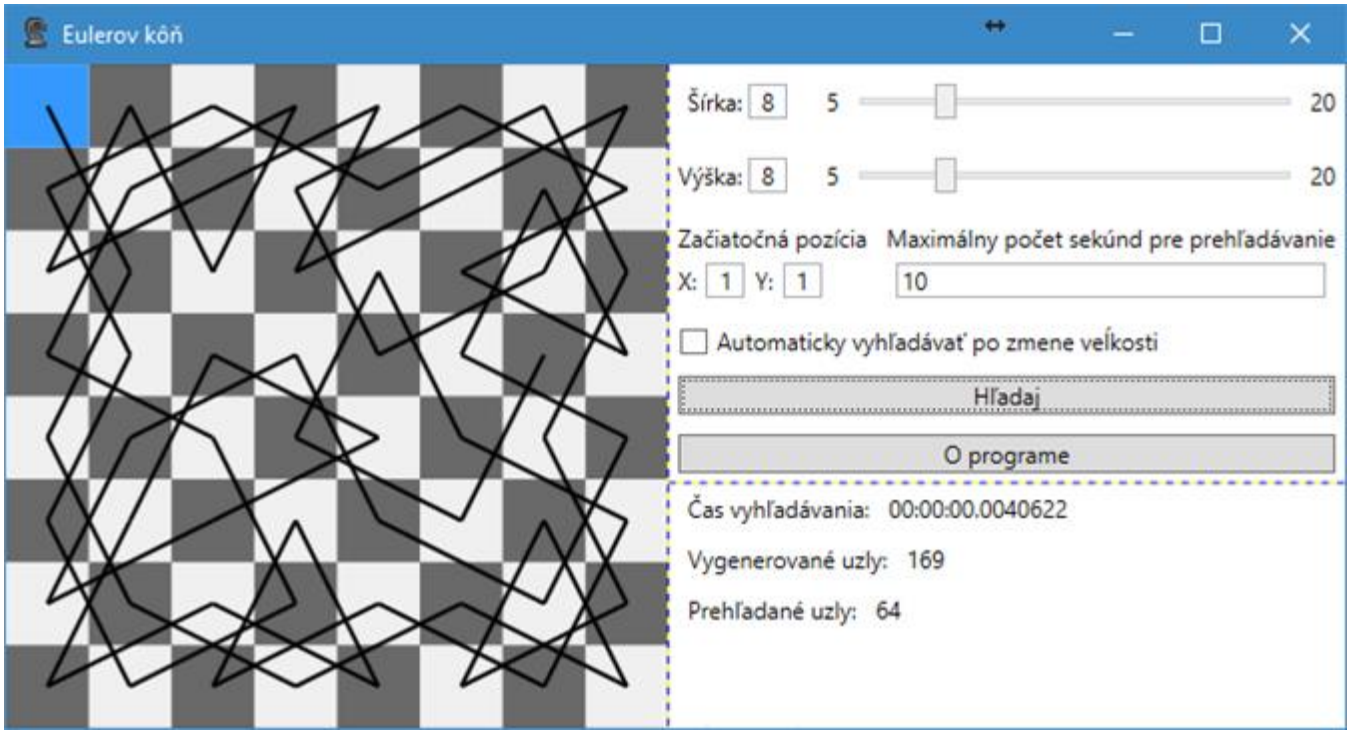
Zhodnotenie

Warnsdorfové pravidlo dokáže výrazne zlepšiť vyhľadávanie. Pre veľkosť šachovnice 20x20 vyhľadávanie trvá v priemere menej než 0.02 sekundy (3,2 GHZ; 64 bit). Po testovaní na rôznych vstupoch sa mi podarilo zistiť, že stále existujú prípady kedy aj za pomoci použitia tejto heuristiky sa dĺžka prehľadávania predĺži aj viac ako 100-násobne. Spomalenie môžeme pozorovať pri úzkych a vysokých rozmeroch šachovnice alebo v niektorých prípadoch kedy je začiatok blízko kraja, keďže vyhľadávanie trvá dlhšie a môže aj presiahnuť časový limit. Jeden s pomalých vstupov je začiatok 5x5 pre rozmery 5x8 alebo začiatok 5x2 pre 8x8.

Algoritmus si v pamäti uchováva všetky stavy (cez spätné referencie) a nespracované uzly. Jeden uzol (a stav) má priemerne 3 KB (Uzol nemá veľa informácií navyše). Pri veľkosti 20x20 v hĺbke 400 je zapamätaných 969 uzlov (čo je zároveň maximum) a 1369 (o 400 viac keďže po každej úrovni sa vyberie uzol a uchová sa iba stav) čo je menej ako 5 MB pre najväčšiu šachovnicu (v prípade, že Garbage Collector stíha odstraňovať stavy, na ktoré sa už

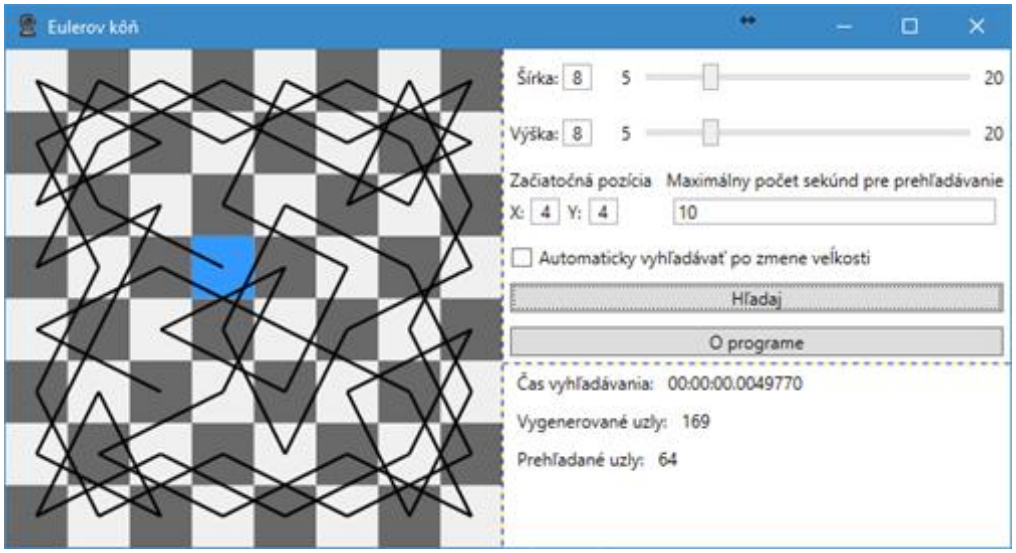
neodkazujeme). Pre rovnaké rozmery v hĺbke 100, pokiaľ sa začínalo v ľavom hornom rohu bude vygenerovaných 355 uzol (stavov) a aktuálne zapamätaných 255 uzlov.

Je ešte možné znížiť využitie pamäte, kedy namiesto zapamätania si celého stavu, by sme si pamätali iba súradnice, na ktorých stál kôň a odkaz na predošlú súradnicu.

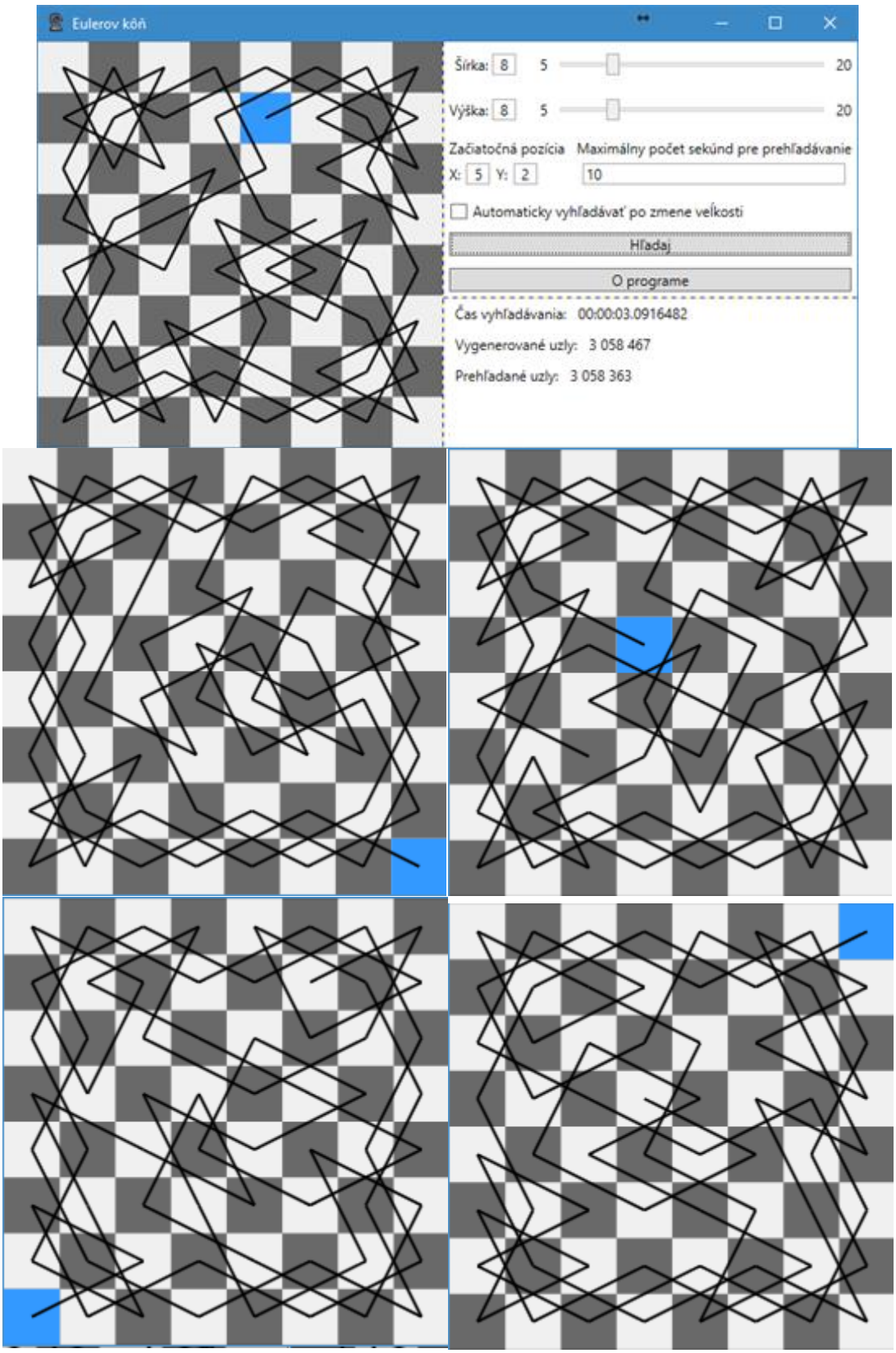


obr.č.2 – Vzhľad aplikácie

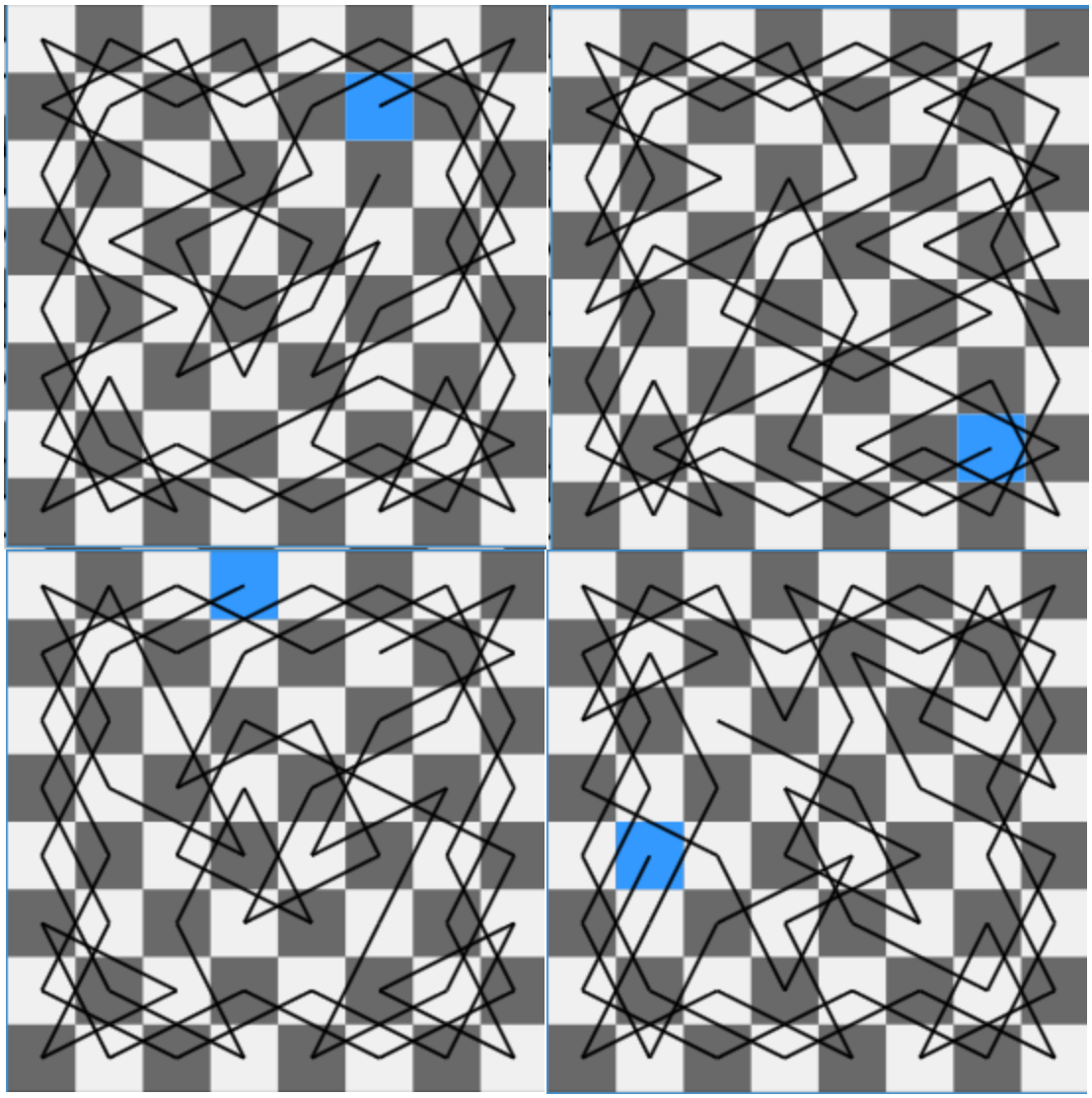
Nájdené riešenia pre šachovnicu 8x8



Matúš Tundér (74288)
Utorok 8:00



Matúš Tundér (74288)
Utorok 8:00



Matúš Tundér (74288)
Utorok 8:00