Slovenská technická univerzita
Fakulta informatiky a informačných technológii
Ilkovičova 3, 812 19 Bratislava

Umelá inteligencia

Zadanie č. 2

Eulerov kôň

Adam Žúrek

Cvičiaci: Mgr. Peter Laurinec
Študijný odbor: Informatika
Ročník: 2. Bc
Akademický rok: 2016/2017

Zadanie

Algoritmom slepého prehľadávania (do hĺbky) je možné nájsť (všetky) riešenia (v bežných výpočtových – čas a pamäť – podmienkach PC) iba pri šachovniciach do veľkosti 6x6, max 7x7. Implementujte tento algoritmus pre šachovnice s rozmermi 5x5 a 6x6 a skúste nájsť prvých 5 riešení pre každú šachovnicu tak, že pre šachovnicu 5x5 aj 6x6 si vyberte náhodne 5 východzích bodov (spolu teda 10 východzích bodov) s tým, že jeden z týchto bodov je (pre každú šachovnicu) ľavý dolný roh a pre každý z týchto bodov nájdite (skúste nájsť) prvé riešenie. V prípade, že ho v stanovenom limite nenájdete, signalizujte neúspešné hľadanie. V diskusii potom analyzujte pozorované výsledky.

Poznámka

Pre problém Eulerovho kôňa treba v oboch úlohách uvažovať s tým, že pre niektoré východzie políčka a niektoré veľkosti šachovnice riešenie neexistuje. Program preto treba navrhnúť a implementovať tak, aby sa v prípade, že do určitého času, resp. počtu krokov riešenie nenájde, zastavil a signalizoval neúspešné hľadanie. Maximálny počet krokov, resp. maximálny čas hľadania by preto mal byť ako jeden zo vstupných (voliteľných) parametrov programu. Pre toto zadanie a testovacie príklady je odporúčaný maximálny počet krokov milión, resp. maximálny čas 15 sekúnd.

Riešenie

Implementáciu danej úlohy som vypracoval pomocou programovacieho jazyka C. Použil štandardnú knižnicu stdio.h na vstup a výstup programu a knižnicu stdbool.h, za pomocou ktorej som si do programu pridal premenné typu boolean.

Hlavnou myšlienkou môjho programu bolo rekurzívne sa vnárať do všetkých ôsmich smerov, ktorými sa mohol kôň pohnúť a hľadať cestu, po ktorej môže pokračovať ďalej. Ak už sa kôň nemôže posunúť ďalej (už prejdené políčko alebo políčko mimo stanovený rozmer) vetva sa ukonči a funkcia vráti hodnotu false. Ak sa počet krokov koňa rovná počtu všetkých políčok, funkcia vráti true a vypíše sa postup cesty koňa.

Ako ohraničenie výpočtovej náročnosti programu som zadefinoval limit milión krokov v programe. Ak bude riešenie vyžadovať väčší počet krokov ako milión, program zastaví proces a upozorní používateľa.

Reprezentácia údajov problému

Hraciu plochu (šachovnicu) som reprezentoval dvojrozmernou maticou celočíselných premenných typu integer int chessBoard[n][n], ktorú program nazačiatku behu inicializuje na hodnotu -1. Ako bude kôň postupovať a prehľadávať voľné políčka, program bude meniť hodnotu prehľadaných políčok na hodnotu poradia kroku, ktorý práve kôň uskutočnil.

Súradnice koňa môžu byť užívateľom zadané pri volaní funkcie a sú to int hose_x a int horse_y.

Ako reprezentáciu pohybov som si vytvoril dve polia int xMove[MOVES] a int yMove[MOVES], kde konštanta MOVES reprezentuje počet možných ťahov, v tomto prípade osem. Hodnoty v poli reprezentujú počet posunutí koňa po x-ovej a y-ovej osi na hracej ploche (matici).

Použitý algoritmus

Použil som algoritmus slepého prehľadávania do hĺbky (backtracking). Jeho podstatou je hľadanie prvého možného riešenia problému. Tento problém som vypracoval za pomoci rekurzie.

Spôsob testovania

Na testovanie funkčnosti programu som použil ako rôzne veľkosti šachovnice, tak aj rôzne začiatočné polohy koňa.

Veľkosti šachovnice som volil od rozmeru 5x5 po rozmer 8x8, pričom rozmer 8x8 nezbehol v stanovenom ohraničenom počte krokov. Väčšie rozmery som už neskúšal, pretože by to bolo zbytočné. Nájdenie cesty s rozmermi 6x6 zaberiem pomerne viac času ako s rozmermi 5x5. Na niektoré šachovnice 6x6 mi nestačil stanovený počet krokov a na niektorých 5x5 maticiach program nevedel nájsť správnu cestu z dôvodu zle vybranej začiatočnej polohy koňa.

Automatické vykonanie programu pre 10 rôznych testovacích vstupov je implementované v programe test.c.

Samostatný program s vlastnými vstupmi je dostupný po spustení euler.c.

Zhodnotenie riešenia

Zadané riešenie je veľmi jednoducho porozumiteľné a implementovateľné. Jeho veľkou nevýhodou je časová náročnosť a obmedzenie veľkosti šachovnice. V mojom programe som neriešil ošetrenie vstupov od používateľa (predpokladám, že používateľ je s problematikou oboznámený) a takisto si používateľ nedokáže sám navoliť časové ohraničenie programu, no môže zmeniť hodnotu konštanty LIMIT, ktorej hodnota reprezentuje počet krokov programu.