Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Umelá Inteligencia

**Zadanie č.1**

**g) Eulerov kôň s Warnnsdorfovou heuristikou**

Akademický rok 2022/2023

**Meno:** Ján Ágh **Dátum:** 27.9.2022

**Cvičiaci:** Ing. Martin Komák, PhD. **Počet strán:** 7

**Obsah**

[**1 Stručný opis problematiky a zadania** 1](#_Toc115627118)

[**2 Navrhnuté riešenie** 2](#_Toc115627119)

[**2.1 Použité programové prostriedky** 2](#_Toc115627120)

[**2.2 Organizácia projektu** 2](#_Toc115627121)

[**2.3 Opis a diagram algoritmu prehľadávania** 3](#_Toc115627122)

[**2.4 Možné výsledky prehľadávania** 5](#_Toc115627123)

# **1 Stručný opis problematiky a zadania**

Cieľom úlohy je navrhnúť program, ktorý dokáže prejsť šachovnicu ľubovoľnej veľkosti väčšej alebo rovnej 5x5 (minimálny rozsah 5x5 až 20x20) ťahmi šachového koňa spôsobom, aby bolo každé políčko šachovnice navštívené práve raz, a zároveň poskytuje používateľovi možnosť zvoliť si ľubovoľnú začiatočnú pozíciu. Do programu je potrebné zapracovať takzvanú Warnnsdorfovu heuristiku, ktorej princíp spočíva v tom, že sa šachový kôň vždy posunie na políčko, odkiaľ má najmenej možností posunúť sa ďalej. V ďalšom kroku je potrebné nájsť pre 10 rôznych východzích pozícií na šachovnici rozmerov 8x8 jedno správne riešenie.

# **2 Navrhnuté riešenie**

## **2.1 Použité programové prostriedky**

Na riešenie úlohy bol použitý programovací jazyk C# v spojení s .NET framework verzie 6.0.300. Riešenie bolo vyvíjané v prostredí Visual Studio Code verzie 1.72.1 a pri jeho vývoji boli využité základné objektovo orientované princípy spolu s návrhovým vzorom Singleton.

## **2.2 Organizácia projektu**

Implementácia sa nachádza v adresári s názvom “*src*”. Tento adresár obsahuje hlavné súbory *Program.cs* a *EulerHorse.csproj*, slúžiace na spustenie vykonávania programu, a trojicu ďalších adresárov *constants, logic*a *models*.

Adresár *constants* obsahuje jediný súbor *Translations.cs* so všetkými ôsmimi dovolenými krokmi šachovnicového koňa.

Adresár *models* obsahuje objektovú reprezentáciu hlavných prvkov programu. Nachádzajú sa tu triedy *CheckerBoard.cs*predstavujúci šachovnicu ako takú, *Square.cs* reprezentujúci jedno políčko šachovnice, *Knight.cs* predstavujúci šachovnicového koňa a nakoniec *Coordinates.cs*, z ktorého dedia *Knight.cs* aj *Square.cs* a ktorý obsahuje horizontálnu a vertikálnu súradnicu prvkov (či už statickú (nemeniacu sa) pozíciu štvorca na šachovnici alebo neustále sa meniacu pozíciu šachovnicového koňa).

Posledný a zároveň hlavný adresár *logic* obsahuje jadro programu spolu s pomocnými funkcionalitami. Súbor *SolvingLoop.cs* slúži na nájdenie riešenia problému s využitím Warnnsdorfovej heuristiky a rekurzie, statická trieda *Validation.cs* obsahuje metódy na validáciu pohybu šachovnicového koňa a posledná trieda *Counter.cs* predstavuje nástroj na meranie času vykonávania programu v milisekundách s presnosťou troch desatinných miest.

Graphical user interface, text, application

Description automatically generated

*Obr. 2.1 Adresárová štruktúra implementácie*

## **2.3 Opis a diagram algoritmu prehľadávania**

Diagram

Description automatically generated

*Obr. 2.2 Vývojový diagram rekurzívneho hľadania správnej cesty*

Logika obsiahnutá v slučke rekurzívneho prehľadávania je relatívne jednoduchá. Na začiatku každého volania metódy *SolveNextStep()* sa zistí, či už nebola vyplnená posledná prázdna pozícia šachovnice a, ak ešte nebola, či už neuplynul časový limit 20 sekúnd.

Text

Description automatically generated

*Obr. 2.3 Overenie, či má zmysel pokračovať v hľadaní (metóda SolveNextStep() triedy SolvingLoop)*

Text

Description automatically generatedV prípade, ak časový limit ešte neuplynul, v ďalšom kroku sa pomocou metódy *RankTranslations()* zvolia všetky ťahy, pomocou ktorých sa kôň dokáže presunúť z jeho súčasnej pozície na novú, ešte nenavštívenú pozíciu.

*Obr. 2.4 Metóda RankTranslations() triedy SolvingLoop*

V tejto metóde sa iteruje cez všetky existujúce kroky pohybu a pomocou metód z triedy *Validations.cs* sa zisťuje, či je možné daný krok vykonať. Ak ide o legálny krok, pomocou metódy *GetOnwardMovesFromSquare()* triedy *CheckerBoard.cs* sa začne uplatňovať Warnnsdorfova heuristika – táto metóda totižto zisťuje váhu kroku (t.j. počet legálnych krokov, ktoré sú z tejto novej pozície uskutočniteľné) a podľa tejto váhy sú jednotlivé legálny kroky v zozname zoradené vzostupne.

Text

Description automatically generated

*Obr. 2.5 Metóda GetOnwardMovesFromSquare triedy CheckerBoard*

Akonáhle má metóda *SolveNextStep()* k dispozícii všetky legálne kroky pohybu vo vzostupne zoradenom zozname, zvolí sa prvý krok (s najnižšou cenou), kôň sa pomocou tohto kroku presunie na novú pozíciu, pozícia sa označí (tieto údaje sú uložené v inštancii koňa, resp. štvorca šachovnice) a začne sa rekurzívne volanie *SolveNextStep()*, pričom toto volanie pokračuje buď dovtedy, kým nie je splnená jedna z podmienok na Obr. 2.3, alebo sa prehľadajú všetky možnosti a zistí sa, že pre daný vstup neexistuje riešenie.

Text

Description automatically generated

*Obr. 2.6 Rekurzívna časť metódy SolveNextStep()*

## **2.4 Možné výsledky prehľadávania**

Počas vykonávania prehľadávania môžu nastať tri situácie, kvoli ktorým sa hľadanie ukončí. Najžiadúcejšia z nich nastane, ak je splnená prvá podmienka na Obr. 2.3, t.j. program našiel riešenie pre daný vstup a rekurzia sa ukončila. V tomto bode každé rekurzívne volanie postupne vráti hodnotu “true”. Druhá situácia nastane, keď vypršal stanovený časový limit a program nestihol nájsť riešenie, prípadne riešenie neexistuje. Vtedy bude splnená druhá podmienka na Obr. 2.3 v poslednom rekurzívnom volaní a pri každom predchádzajucom bude splnená podmienka v “else if” vetve na Obr. 2.6, t.j. postupne sa bude vraciať hodnota “false”, až kým program neskončí. Posledná, tretia situácia sa stane realitou, keď metóda *RankTranslations()* vráti prázdny zoznam krokov, čo značí, že sa kôň ocitol v slepej uličke a nemá sa kam posunúť. V tomto bode sa vráti hodnota “false, v predchádzajúcom rekurzívnom volaní sa vykoná vetva “else” na Obr. 2.6, odznačí sa daný štvorec šachovnice a kôň sa posunie späť na predchádzajúcu pozíciu. Keď sú tieto kroky vykonané, začne ďalšia iterácia v cykle foreach (z utriedeného zoznamu krokov sa vyberie nasledujúci krok) a hľadanie pokračuje ďalej dovtedy, kým sa cesta nenájde, nevyčerpajú sa všetky možnosti v každom rekurzívnom volaní alebo nevyprší časový limit.