Peter Kúdela

UI

Zadanie 2

8-hlavolam

A\* algoritmus

28.10.2020

**Používanie**

stateGen.py

V adresári sa nachádza program stateGen.py ktorý slúži na vygenerovanie predpripravených vstupov (vstupy sú generované vo forme txt súborov v adresári kde bol program spustený).

main.py

Po spustení bude užívateľ vyzvaný vložiť názvy súborov reprezentujúcich začiatočný a konečný stav hlavolamu, nasledovne výber heuristiky. Po týchto voľbách program vypíše graficky kroky postupu, trvanie priebehu, počet spracovaných a vytvorených uzlov a postupnosť krokov riešenia.

**Štruktúra programu**

Node

Node je reprezentácia uzlu, ktorá si pamätá stav hlavolamu ako aj nasledovné informácie:

*Grid* – reprezentácia stavu v tvare 2D poľa

*Parent* – odkaz na rodičovský uzol

*Depth* – hĺbka uzla potrebná pre A\* (hodnota q)

*H* – výsledok heuristickej funkcie

*F* – depth + h potrebná pre A\*

*lastOp* – operácia ktorou sme sa na tento stav dostali (zmenšuje počet vytvorených potomkov čo znižuje čas hľadania)

A taktiež pre node špecifické funkcie:

*Shifts* – mená funkcií korešpondujú so smerom posunu

*showGrid* – vypíše 2D pole

*heur* – odkaz na užívateľom zvolenú heuristiku

*heur1* – implementácia heuristiky 1

*heur2* – implementácia heuristiky 2

Dictionaries opened a closed

Opened a closed slúžia na ukladanie vygenerovaných nespracovaných a spracovaných stavov. Tvoria ich kombinácie uzlov a ich kľúčových hodnôt (za kľúčové hodnoty som vybral výsledok heuristickej funkcie čo mi pomáha znížiť čas pri hľadaní výskytu stavov v denníkoch)

Pomocné funkcie

Okrem funkcii objektu node a samotného algoritmu sa v programe nachádzajú nasledovné pomocné funkcie:

*findNum* – ak sa v 2D poli hľadané číslo nachádza, vráti súradnice x a y

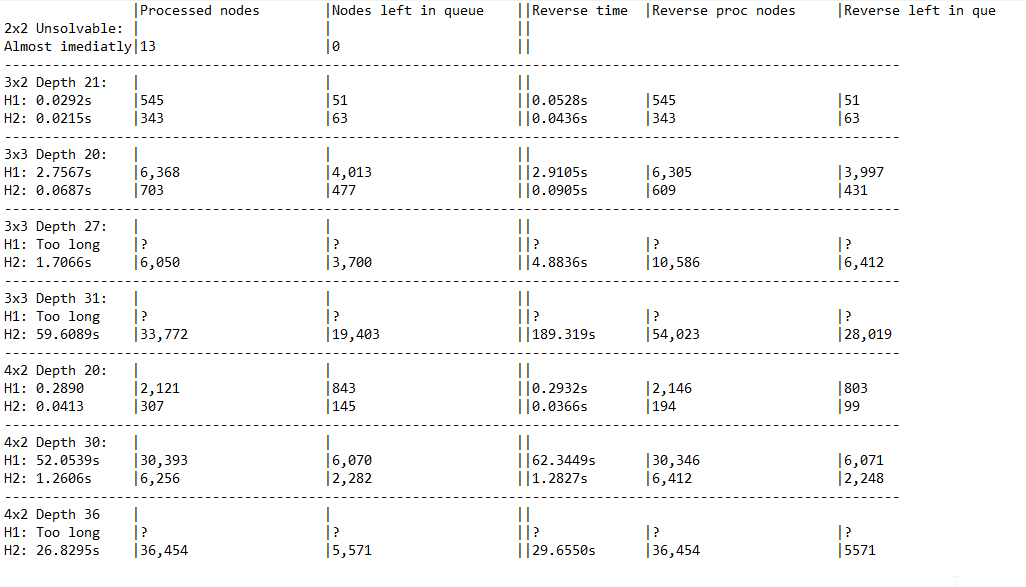
*showPath* – vykreslí na výstup cestu k danému uzlu

*isIn* – vráti True ak sa v denníku nachádza uzol s rovnakým stavom ktorého hodnota f je menšia

**A\* algoritmus**

Algoritmus prebieha v cykle ktorý sa vykonáva kým sú uzli v denníku opened. Na začiatku každej iterácie vyberieme z denníku opened uzol s najmenšou hodnotou f, tento uzol nasledovne z denníku odstránime. Ďalej vytvoríme 3-och potomkov súčasného uzla (okrem spätného, môžu byť 4 ak ide o počiatočný stav), potomkov ktorým sa nepodarí vykonať shift operácia, alebo sa ich stav nachádza v denníkoch opened alebo closed s menšou hodnotou f ignorujeme. Na konci iterácie pridáme súčasný uzol do denníku closed. Kontrolu či sme našli cieľový stav vykonávame pri tvorbe potomkov, ak ho nájdeme vypíšeme cestu. Ak nám neostanú žiadne uzli v opened, funkcia končí a vypisuje, že hlavolam nemá riešenie.

**Testovanie**

Program bol testovaný na 8-mich vstupoch, jeden z nich je nesplniteľný (2x2, hlavolamy väčších rozmerov by trvali príliš dlho) všetky s použitím oboch heuristík. Výsledky vypadali nasledovne:

H1 je heuristika, súčet kameňov na nesprávnom mieste, H2 reprezentuje súčet vzdialeností od cieľových pozícií. Všetky tri zaznamenané údaje kontrolujeme aj pri hľadaní opačným smerom.

**Analýza testovania**

V mojej implementácií je druhá heuristika omnoho rýchlejšia nie len na základe toho, že je presnejšia, ale aj kvôli spôsobu adresovania v programe, kde výsledok heuristiky slúži ako kľúčová hodnota v denníkoch. Pri vytváraní programu som originálne používal obyčajné polia, pri porovnaní heuristík bola druhá stále rýchlejšia no rozdiel neboľ až tak výrazný (starú verziu programu som nechal v adresári misc).

Pri prehodení začiatočných a konečných stavov sa výsledky zväčša zhoršili.