2. zadanie z predmetu ISI – Lights out - Štucka, Telepčák

DFS:

Dfs algoritmus, algoritmus pre vyhľadávanie riešenia do hĺbky, pracuje pre naše zadanie nasledovným spôsobom. Hlavný cyklus pozostáva z inicializácie stackov potrebných pre evidovanie počiatočného stavu, otvorených stavov a navštívených stavov. Pred začatím cyklu vložíme do stacku initial state, ktorý budeme vlastne ďalej expandovať. Cyklus teda beží kým je v stacku nejaký stav. Ak už nie je, boli všetky prejdené a nemá riešenie. V cykle teda vyhodíme stav zo stacku ale necháme si ho v bočnej premennej. Nasleduje kontrola, či aktuálny stav nie je finálny stav. Ak áno, nastavíme cestu k tomu stavu , „zhasneme všetky svetlá“ a uchovávame počet krokov. Ak to nie je finálny stav, práca cyklu pokračuje, a to tak, že ďalej kontroluje či už tento stav nebol navštívený (pri uchovávaní stavov, sa uchováva aj „explored“ hodnota). Ak je to nový stav, nastavíme ho ako navštívený, pushneme do stacku s navštívenými stavmi, a zatiaľ si uchovávame aktuálnu cestu ku stavu. Pre všetky políčka herného poľa vykonáme možnosti ďalších stavov pri kliku na nich. Volaná funkcia vytvára stavy ktoré reprezentujú v tvare + zmenu stavu políčok. Kontrolujeme, či už nie sú objavené, alebo dokonca aj otvorené tieto nové stavy. Ak nie, zapíšeme nové stavy do otvorených stavov, ktoré sa neskôr budú expandovať pre ďalšiu iteráciu v cykle. Ak skončí cyklus preto lebo má prázdny stack, znamená to, že hra nemá riešenie. Ak by mala, v cykle by už toto riešenie bolo označené.

Greedy:

Greedy teda pridá na začiatku vstupný stav do stacku. Ďalej už nasleduje cyklus ktorý rovnako ako DFS ide kým nie je prázdny stack s otvorenými stavmi. Nasledujúci stav sa vyberá zo všetkých otvorených stavov na základe heuristiky, ako boli výhodnostne ohodnotené otvorené stavy, vyberie sa teda najlepší. Algoritmus následne skontroluje, či vybraný stav nie je finálny stav. Ak áno, vyfarbí herné pole a ukončí cyklus a vypíše detaily cyklu. Ak nie, pokračuje ďalej vyhodením stavu zo stacku aby sa teda neopakoval. Pre každé políčko sa nájde možný ďalší stav, kontroluje či už bol expandovaný a keď už bol v stavoch na expandovanie, aktualizuje hodnotu z heuristiky, ak nebol v stacku, priloží ho tam. Cyklus ide, kým sú v stacku stavy na expandovanie a nasledujúci stav sa vyberie podľa heuristiky ktorá vlastne vyhodnocuje výhodnosť stavu na základe rozsvietených „svetiel“

A*:

Znova, inicializácia potrebných stackov. Vložíme vstupný štartovací stav do stacku. Nasleduje cyklus, opakuje sa kým je v stacku stav na expandovanie. Program ďalej vyberie najlepší stav, najvýhodnejšie ohodnotený heuristikou. Heuristika vlastne pozostáva zo súčtu vzdialenosti stavu v grafe a počtu „zasvietených“ políčok. Skontroluje či stav už je finálny, ak nie, pokračuje vyhodením stavu z open states stacku aby sa neopakoval. Nasleduje pre každé políčko simuluje kliknutie a nové stavy ak neboli ešte expandované pridajú do stacku , ohodnotené heuristikou alebo aktualizujú heuristiku pre daný stav. Cyklus sa opakuje stále kým je splnená podmienka.

			DFS	GREEDY	A*
3 x 2	1. Mapa	ČAS	**0.014 s**	**0.014 s**	**0.009 s**
		POČET KROKOV	**11**	**9**	**5**
	2. Mapa	ČAS	**0.18 s**	**0.006 s**	**0.006 s**
		POČET KROKOV	**14**	**1**	**1**
	3. Mapa	ČAS	**0.014 s**	**0.009 s**	**0.044 s**
		POČET KROKOV	**10**	**5**	**5**
5 x 5	1. Mapa	ČAS	**0.03 s**	**0.053 s**	**0.102 s**
		POČET KROKOV	**10**	**9**	**29**
	2. Mapa	ČAS	**-**	**0.024 s**	**0.012 s**
		POČET KROKOV	**Veľmi veľa**	**6**	**4**
	3. Mapa	ČAS	**-**	**100.777 s**	**0.016 s**
		POČET KROKOV	**Veľmi veľa**	**1030**	**7**

Tabuľka 1: Dosiahnuté výsledky jednotlivých algoritmov