

**Lights Out**  
**Zadanie č.2**

**Erik Laki a Patrik Čmil**

## Riešenie Lights Out použitím DFS algoritmu

DFS funguje na princípe LIFO (Last In First Out). Posledný stav, ktorý pridáme do stacku tak vlastne aj popneme a budeme ho opäť rozširovať do hĺbky, čo najhlbšie, až kým nenarazíme na to, že nebudeme môcť ďalší stav expandovať. V tom prípade sa vrátíme späť a skúsime iné vetvy, ktoré ešte neboli navštívené.

Na začiatku máme 2 prázdne polia **stack** a **visited**.

1. Pridáme počiatočný stav, to znamená v prípade hry Lights Out dvojrozmerné pole s hodnotami **true alebo false**, do stacku a do visited listu a tak isto nastavíme rodiča tohto stavu na **None**, keďže je počiatočný.
2. Následne sa začína while cyklus a nasledujúce kroky sa opakujú až kým stack nebude prázdny:
  - a. Do premennej `current_state` si **popneme stav zo stacku**.
  - b. Pridáme ho do visited poľa, v ktorom uchovávame stavy, ktoré už boli navštívené.
  - c. Zistujeme či aktuálny stav, t.j. `current_state` je konečný stav(všetky svetlá sú vypnuté).
    - i. Ak áno tak z aktuálneho stavu prechádzame **parents** slovník, v ktorom uchovávame rodičov pre každý stav.
    - ii. Pridávame rodiča aktuálneho stavu do **moves listu**.
    - iii. Aktuálny stav nastavíme na rodiča, to aby sme sa dostali až k počiatočnému stavu.
    - iv. **Moves list** otočíme pomocou funkcie **reverse** aby sme dostali jednotlivé movy od počiatočného až po koncový stav.
  - d. Generujeme všetkých potomkov aktuálneho stav, v prípade našej hry to znamená, že postupne stlačíme každé jedno políčko v našom poli.
    - i. **Generujem** potomkov postupne vo **for cykle**.
    - ii. Vždy keď vygenerujem potomka tak zistím či sa už nachádza vo visited liste
    - iii. Ak sa **nenachádza** tak ho pridáme do **stacku** a do **visited** listu a tak isto mu nastavíme ako rodiča, aktuálny stav
    - iv. Toto robíme pre všetkých potomkov aktuálneho stavu

## Riešenie Lights Out použitím Greedy Search algoritmu

Greedy search funguje tak, že na každom kroku sa snaží vybrať najlepšiu cestu v aktuálnom momente. V našom prípade to bude, že si bude vyberať ako nasledujúci krok stav, ktorý bude mať najnižšiu heuristiku.

Heuristika, ktorú sme si vybrali v našom zadaní je **počet svetiel ,ktoré svietia**.

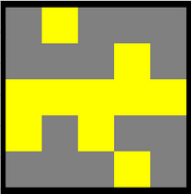
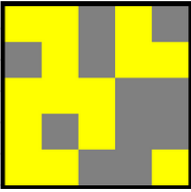
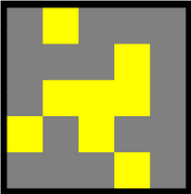


Keďže náš cieľový stav pre ľubovoľné pole je to aby boli všetky svetlá **zhasnuté**.

Budeme používať **priority stack**, z ktorého si vlastne **popnem state** , ktorý má najnižšiu **hodnotu heuristiky** a ten budem ďalej rozširovať.

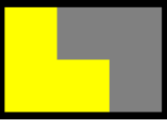




1. Inicializujem si **priority\_stack** a **closed list** a tak isto **parents slovník**
2. Pridám počiatočný stav do **priority\_queue** dvojicu hodnôt **heuristiku aktuálneho stavu a aktuálny stav**
3. A tak isto si pridám počiatočný stav aj do **closed listu** a nastavím rodiča počiatočného stavu na None keďže žiadneho nemá
4. Nasledujúce kroky opakujem kým naša **priority queue nie je prázdna**
  - a. Aktuálny stav = stav **popnutý z priority\_queue s najnižšou hodnotou heuristiky**
  - b. Zistíme či tento stav je náš cieľový stav
    - i. Ak áno tak opakujeme rovnaký postup ako pri DFS algoritme
  - c. Opäť si vygenerujeme potomkov aktuálneho stavu a prechádzame ich vo for cykle
    - i. Ak **potomok** sa nenachádza **v closed liste** tak ho pridáme do **priority queue** ako **dvojicu jeho heuristiky a je samotného stavu**
    - ii. Tak isto ho priadme aj do closed listu a nastavíme jeho rodiča na aktuálny stav z ktorého sme týchto potomkov získali

# Štatistiky

## 5x5 polia

<b>Algoritmus</b>	<b>Pocet expandovanych stavov</b>	<b>Pocet ťahov v riešení</b>	<b>Čas behu algoritmu(sekundy)</b>		Hracie pole : č.0	
DFS	8388602	242934	1178,94			
Greedy Search	20870	42	0,33			
A*						
<b>Algoritmus</b>	<b>Pocet expandovanych stavov</b>	<b>Pocet ťahov v riešení</b>	<b>Čas behu algoritmu(sekundy)</b>		Hracie pole : č.1	
DFS	8225071	891191	342,5			
Greedy Search	8394	53	0,13			
A*						
<b>Algoritmus</b>	<b>Pocet expandovanych stavov</b>	<b>Pocet ťahov v riešení</b>	<b>Čas behu algoritmu(sekundy)</b>		Hracie pole : č.2	
DFS	8388609	113411	1731,11			
Greedy Search	12978	29	0,19			
A*						
<b>Algoritmus</b>	<b>Pocet expandovanych stavov</b>	<b>Pocet ťahov v riešení</b>	<b>Čas behu algoritmu(sekundy)</b>		Hracie pole : č.3	
DFS	8388607	216512	1348,93			
Greedy Search	3375	28	0,05			
A*						
<b>Algoritmus</b>	<b>Pocet expandovanych stavov</b>	<b>Pocet ťahov v riešení</b>	<b>Čas behu algoritmu(sekundy)</b>		Hracie pole : č.4	
DFS	8338905	735650	728,6			
Greedy Search	5897	30	0,08			
A*						

## 2x3 polia

Algoritmus	Pocet expandovanych stavov	Pocet tahov v riešení	Čas behu algoritmu(sekundy)		Hracie pole : č.0	
DFS	17	1	0,0009			
Greedy Search	7	1	0			
A*						
Algoritmus	Pocet expandovanych stavov	Pocet tahov v riešení	Čas behu algoritmu(sekundy)		Hracie pole : č.1	
DFS	17	2	0			
Greedy Search	12	2	0			
A*						
Algoritmus	Pocet expandovanych stavov	Pocet tahov v riešení	Čas behu algoritmu(sekundy)		Hracie pole : č.2	
DFS	17	2	0			
Greedy Search	14	2	0			
A*						
Algoritmus	Pocet expandovanych stavov	Pocet tahov v riešení	Čas behu algoritmu(sekundy)		Hracie pole : č.3	
DFS	17	2	0			
Greedy Search	14	2	0			
A*						
Algoritmus	Pocet expandovanych stavov	Pocet tahov v riešení	Čas behu algoritmu(sekundy)		Hracie pole : č.4	
DFS	17	1	0,001			
Greedy Search	7	1	0,001			
A*						