**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ**

# Zadanie 2

# 8-HLAVOLAM S VYUŽITÍM A* ALGORITMU

Anna Yuová

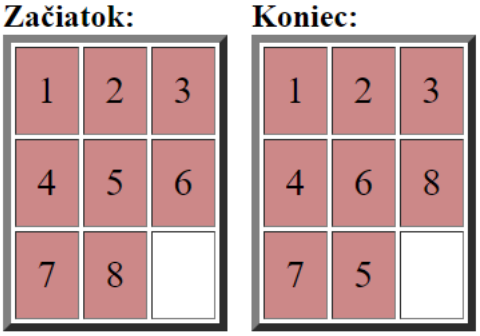
**Predmet:** Umelá inteligencia

**Akademický rok:** 2020/2021

**Semester:** zimný

# ZADANIE ÚLOHY

Úlohou je nájsť riešenie 8-hlavolamu. Hlavolam je zložený z 8 očíslovaných políčok a jedného prázdneho miesta. Políčka je možné presúvať hore, dole, vľavo alebo vpravo, ale len ak je tým smerom medzera. Je vždy daná nejaká východisková a nejaká cieľová pozícia a je potrebné nájsť postupnosť krokov, ktoré vedú z jednej pozície do druhej.
Príkladom môže byť nasledovná začiatočná a koncová pozícia:




**Úloha E)**
Použite A* algoritmus, porovnajte výsledky heuristík 1. a 2.

1. Počet políčok, ktoré nie sú na svojom mieste.
2. Súčet vzdialeností jednotlivých políčok od ich cieľovej pozície.

**Postup A* algoritmu podľa zadania je nasledovný:**

1. Vytvor počiatočný uzol a umiestni medzi vytvorené a zatiaľ nespracované uzly
2. Ak neexistuje žiadny vytvorený a zatiaľ nespracovaný uzol, skonči s neúspechom - riešenie neexistuje
3. Vyber najvhodnejší uzol z vytvorených a zatiaľ nespracovaných, označ ho aktuálny
4. Ak tento uzol predstavuje cieľový stav, skonči s úspechom - vypíš riešenie
5. Vytvor nasledovníkov aktuálneho uzla a zaraď ho medzi spracované uzly
6. Vytrieď nasledovníkov a ulož ich medzi vytvorené a zatiaľ nespracované
7. Choď na krok 2.

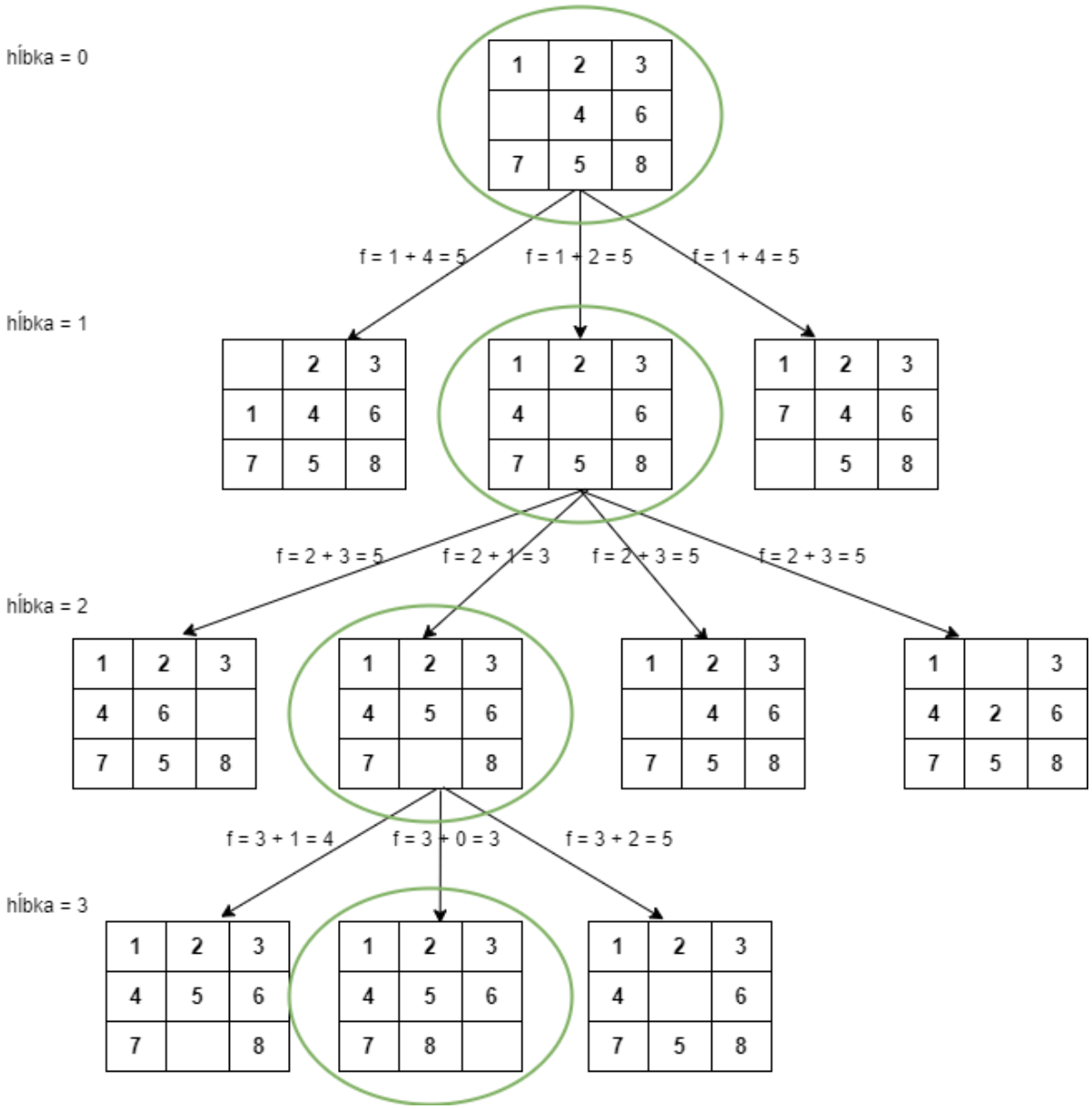
# OPIS RIEŠENIA

Stav môjho hlavolamu je reprezentovaný dvojrozmerným poľom celých čísel, pričom medzera je reprezentovaná najvyšším číslom.
Napríklad pre hlavolam, ktorý má rozmer 3x3 sú pozície reprezentované číslami 1 až 8 a medzera je číslo 9.

Na vyriešenie hlavolamu som podľa zadania použila A* algoritmus a 2 heuristiky – manhattan a hamming.
Na začiatku som si načítala zo vstupu počet riadkov a stĺpcov (m, n) a počiatočné pole. Toto pole som si uložila ako počiatočný uzol – nemá žiadneho predchodcu a jeho hĺbka je 0. Ďalej som v cykle generovala ďalšie stavy, do ktorých sa dá dostať z tohto počiatočného uzlu. Z funkcie si vrátim všetky ďalšie možné stavy, do ktorých sa dá dostať – vie vrátiť minimálne 2 stavy a maximálne 4. Ak je medzera v rohu, vie sa posunúť do 2 smerov a vráti 2 stavy. Ak je medzera na kraji ale nie v rohu, vie sa posunúť do 3 smerov a vráti mi 3 stavy a ak je

medzera hocikde inde (v strede), tak sa vie posunúť do všetkých smerov (hore, dole, doľava, doprava) a vráti mi 4 stavy. Z týchto stavov vyberám iba 1 stav, ktorý má najnižšiu hodnotu f = g(hĺbka) + h(heuristika). Mám 2 heuristické funkcie, cez ktoré viem dostať f. Prvá funkcia je manhattan, ktorá mi vráti súčet pozícii políčok o koľko sú vzdialené od svojej cieľovej pozície. Druhá funkcia je hamming, ktorá mi vráti počet políčok v hlavolame, ktoré nie sú na svojom mieste. Medzera sa neráta ani v jednom prípade. Výsledne pole s najmenšou hodnotou f, som si uložila do výsledného pola, kde mám všetky výsledné uzly a vďaka atribútu predchádzajúci sa k ním viem postupne dopracovať. Ďalej si nastavím to pole s najmenšou hodnotou f, na aktuálny uzol, ktorý idem spracovať a opakujem v cykle, až kým sa to pole nerovná zoradenému hlavolamu, kde na poslednom mieste miesto medzery je najvyššie číslo.

Na obrázku som znázornila ako som postupne prechádzala ďalšie stavy a zelené sú tie, ktoré som vybrala pre najmenšiu hodnotu f a uložila si do výsledného pola, z ktorého sa viem postupne dostať až na začiatok cez predchádzajúceho. Pri každom generovaní nových stavov som zvyšovala hĺbku o 1.



# POPIS A* ALGORITMU

1) Načítanie dvojrozmerného pola zo vstupu

2) Nastavenie pola ako počiatočný uzol a priradenie jeho hodnôt

Cyklus, pokiaľ vygenerovaný stav nie je konečný stav:

3) Z aktuálneho (na začiatku počiatočného) uzla vygenerovanie všetkých ďalších možných stavov, do ktorých sa dá z neho dostať

4) Uloženie všetkých stavov do pola

5) Vybratie jedného stavu zo všetkých možných podľa najmenšej hodnoty f z výpočtu heuristiky

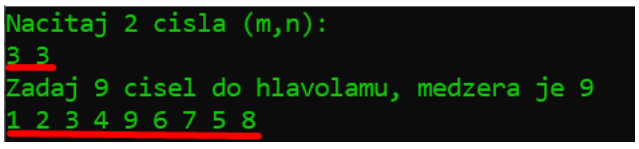
5) Skontrolovanie, či vybraný stav nie je finálny stav – koniec cyklu

6) Ak to nie je finálny stav, nastavenie vybraného stavu na aktuálny stav a uloženie stavu, ktorý bol predtým aktuálny do výsledného poľa stavov

7)Premazanie pola so všetkými stavmi a uvoľnenie pamäte

# TESTOVANIE A SPUSTENIE

Na implementáciu som použila programovací jazyk C a program DEV C++ (verzia 5.11). Program je spustiteľný. Po spustení treba zadať 2 veci – počet riadkov (m) a stĺpcov (n) a potom vstupné pole – veľké m * n.




Testovala som rovnaké vstupy pre obe heuristiky. V prvom riadku tabuľky to zbehlo rýchlo na jednoduchom príklade v oboch prípadoch. Algoritmus nájde správne v oboch heuristikách najkratšiu cestu riešenia. V ďalších prípadoch už bolo vidieť časový rozdiel. Pri heuristike hamming síce vygenerovalo stavy do približne rovnakej alebo aj úplne rovnakej hĺbky, no tých vygenerovaných stavov bolo oveľa viac ako počet vygenerovaných stavov pri heuristike manhattan. Manhattan heuristika mi z toho vyplýva ako časovo efektívnejšia a menej náročná pre pamäť, kedže generuje menej stavov.

V tabuľke som porovnávala výsledky heuristík hamming a manhattan.

| **Počiatočný stav** | **Finálny stav** | **Najväčšia hĺbka pre manhattan** | **Počet vygenerovaných ďalších stavov – heuristika manhattan** | **Najväčšia hĺbka pre hamming** | **Počet vygenerovaných ďalších stavov – heuristika hamming** |
|---|---|---|---|---|---|
| 1 2 3<br>4 9 6<br>7 5 8 | 1 2 3<br>4 5 6<br>7 8 9 | 3 | 10 | 3 | 10 |
| 7 3 2<br>5 9 8<br>4 6 1 | 1 2 3<br>4 5 6<br>7 8 9 | 22 | 1731 | 22 | niekoľkonásobne viac ako pri manhattan |
| 9 7 1<br>8 2 3<br>5 4 6 | 1 2 3<br>4 5 6<br>7 8 9 | 20 | 910 | 20 | 5355 |
| 4 3 1<br>8 9 2<br>7 6 5 | 1 2 3<br>4 5 6<br>7 8 9 | 14 | 102 | 14 | 462 |

Z tabuľky je vidieť, že heuristika manhattan dáva lepšie výsledky a čím je problém zložitejší, tým je väčši rozdiel medzi heuristikami a o to dlhšie to trvá. Manhattan vo všetkých testovaných prípadoch zbehla za pár stotín sekúnd no pri hamming to trvalo oveľa dlhšie. Výhodou manhattan heuristiky je teda časová a pamäťová efektivita oproti hamming. Hamming má tiež výhodu, že ide do približne rovnakej a často rovnakej hĺbky ako manhattan, ale nevýhodou je, že generuje niekoľko násobne viac ďalších možných stavov, do ktorých sa vieme ďalej dostať.

# ZÁVER

Úlohou môjho zadania bolo implementovať A* algoritmus a porovnať výsledky 2 heuristík. Algoritmus našiel vždy najkratšiu cestu k riešeniu s obi dvomi heuristickými funkciami. V porovnaní som zistila, že manhattan heuristika je oveľa efektívnejšia ako hamming, pretože generuje menej ďalších stavov a šetrí čas aj pamäť. Zložitosť tohto algoritmu je O(log n).