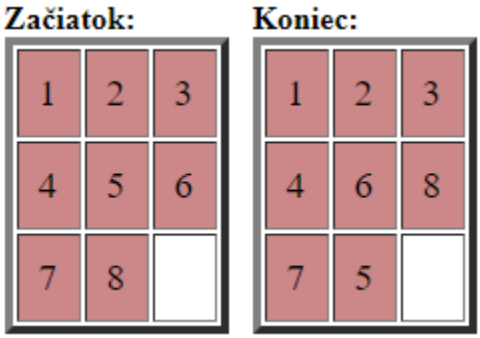
Ľubomír Dlhý, ID: 102897

PREHĽADÁVANIE STAVOVÉHO PRIESTORU

Ľubomír Dlhý, ID: 102897

ZADANIE

Našou 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.



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

RIEŠENIE

Použité heuristiky:
1. Súčet manhattanovských vzdialeností políčok do cieľového stavu
2. Hammingova vzdialenosť (počet zle položených políčok)

Zadaný problém som riešil pomocou A* algoritmu s využitím dvoch heuristík. Obe heuristiky nebrali do úvahy medzeru (číslo 0). Generované uzly boli vkladané do prioritného radu, kde sa prvky vždy usporiadali podľa najmenšej hodnoty funkcie *f*. Funkcia *f* bola počítaná ako súčet funkcií *g* a *h*, kde *g* bola hĺbka cesty a *h* bola heuristická funkcia. Uzol s najmenšou hodnotou sa vždy vybral z prioritného radu a spracoval sa. Spracovať uzol znamená vytvoriť uzol v každom možnom smere – Dole, Hore, Doprava, Doľava – ale iba za predpokladu, že tento pohyb je možné vykonať a že stav uzla ešte nebol spracovaný. Spracované stavy si ukladám v hash tabuľke, aby sa vyvarovalo zacykleniu. Stavy sú reprezentované ako polia znakov o veľkosti m*n, do tabuľky sa vkladajú reťazce. Ak stav vybraného uzla z prioritného radu je rovnaký ako koncový stav, algoritmus sa skončí a vypíše sa cesta. Ak sa postupne vygenerujú a spracujú všetky uzly a nenájde sa uzol s rovnakým stavom ako je koncový stav, pre vstup neexistuje riešenie.

```
uchar manhattan_distance(uchar width, uchar height, char* state, char* dest_state) {
    uchar total = 0;
    for (int i = 0; i < width * height;i++) {
        for (int j = 0; j < width * height; j++) {
            if (state[i] && state[i] == dest_state[j]) {
                total += abs(j/ width - i/width) + abs(j%width -i%width);
                break;
            }
        }
    }
    return total * MKOEF;
}
```

```
uchar manhattan_cached(uchar h, uchar realPosX, uchar realPosY, uchar destPosX, uchar destPosY, char diffX, char diffY) {
    if (diffX == 0)
        return h + MKOEF * (abs(realPosY - destPosY) - abs(realPosY + diffY - destPosY));
    else
        return h + MKOEF * (abs(realPosX - destPosX) - abs(realPosX + diffX - destPosX));
}
```

Pre zefektívnenie programu sa heuristická funkcie počítajú pre každé políčko iba raz – pre začiatočný stav. Pre ďalšie stavy sa hodnota heuristickej funkcie dá vypočítať iba zo zmeny daných políčok. V ukážke je funkcia manhattan_distance, ktorá ráta vzdialenosti pre každé políčko hlavolamu a funkcia manhattan_cached, ktorá ráta vzdialenosť iba z pozmenených políčok a hodnoty heuristickej funkcie z predošlého uzla.

REPREZENTÁCIA STAVOV

Uzol

```
struct vrchol {
    char* state;
    char posZero;
    uchar g;
    uchar h;
    struct vrchol* parent;
};
```

Uzol je implementovaný štruktúrou vrchol, ktorá obsahuje:

- **state** – stav hlavolamu = pole znakov o veľkosti m*n
- **posZero** – pozícia nuly v stave uzla
- **g** – hodnota funkcie g (hĺbky uzly)
- **h** – hodnota funkcie h (heuristiky)
- ***parent** – odkaz na predošlý uzol

Uzol má veľkosť **m*n+7** bajtov, kde **m** a **n** sú rozmery hlavolamu

Prioritný rad a hashovacia tabuľka

```
priority_queue<vrchol*, vector<vrchol*>, CompareNodes> PQ;
unordered_set<string> visited;
```

Prioritný rad (min halda) slúži na to, aby sa spracoval uzol vždy s najmenším ohodnotením, teda s najmenším súčtom hodnôt funkcie *g* a funkcie *h*.
Do hashovacej tabuľky sa ukladajú už vytvorené stavy z dôvodu, aby sa program nezacyklil. Stav sa prekonvertuje do reťazca a potom sa vloží do hashovacej tabuľky.

Ľubomír Dlhý, ID: 102897

ALGORITMUS

A* vyhľadávanie

1. Vytvor prioritný rad a vlož doň štartovací uzol
2. Vytvor hashovaciu tabuľku a vlož do nej stav štartovacieho uzla
3. Pokiaľ nie je prioritný rad prázdny :
   3.1. vyber uzol s najnižším ohodnotením z prioritného radu
   3.2. ak je stav uzla rovnaký ako koncový stav, skonči, inak pokračuj
   3.3. pre každý možný posun (hore, dole, doprava, doľava):
      3.3.1. vytvor nový stav posunom políčok zo starého stavu
      3.3.2. ak nový stav bol už spracovaný, zahoď stav a pokračuj na bod 3, inak vytvor nový uzol
      3.3.3. nastav novému uzlu predchodcu, pozíciu nuly a ohodnotenie
      3.3.4. vlož vrchol do prioritného radu
      3.3.5. vlož stav do hash tabuľky navštívených stavov

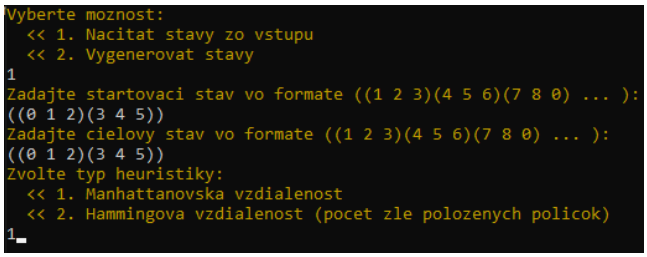
UŽÍVATEĽSKÉ PROSTREDIE

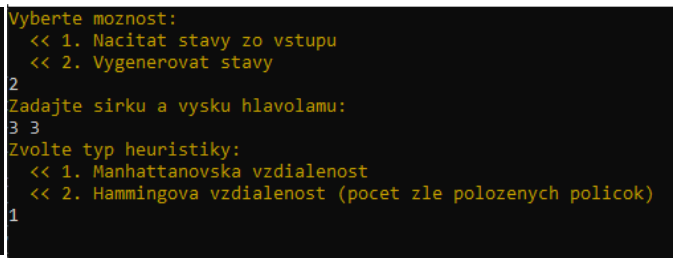
Program je spustiteľný buď cez príkazový riadok s argumentami alebo ako interaktívna konzolová aplikácia.

```
UI_zad.exe "((0 1 2)(3 4 5))" "((3 4 5)(0 1 2))" 1
```

Pri spúšťaní cez príkazový riadok sú potrebné 3 argumenty – začiatočný stav, koncový stav a výber heuristiky. Stavy je potrebné dať do úvodzoviek v hore uvedenom formáte.
Tretí argument, čiže výber heuristiky je buď 1 pre Manhattanovskú vzdialenosť alebo 2 pre Hammingovu vzdialenosť.

Po spustení konzolovej aplikácie sa zobrazí interaktívne menu, kde je možnosť načítať začiatočný a koncový stav zo vstupu od užívateľa alebo vygenerovať náhodné stavy. Pri prvej možnosti treba zadať vstupy v uvedenom formáte, pri druhej možnosti treba zadať rozmery hlavolamu. Nakoniec treba vybrať typ heuristiky.





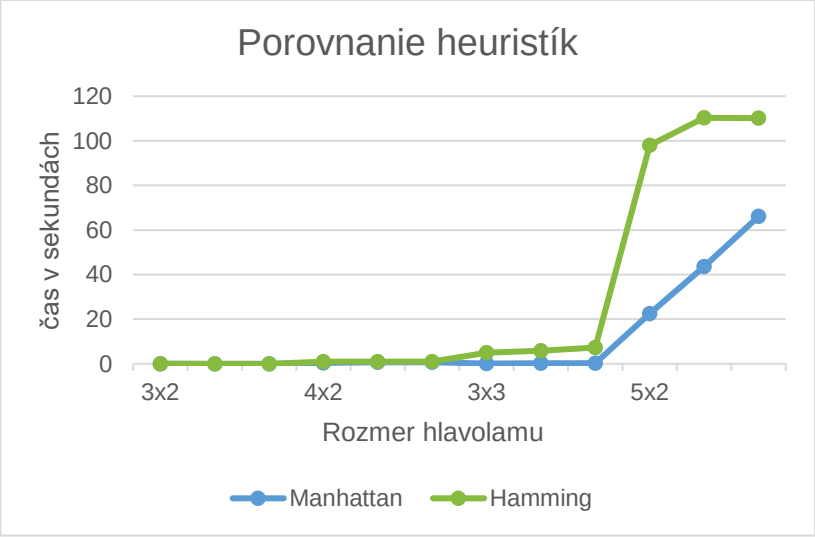
Ľubomír Dlhý, ID: 102897

TESTOVANIE

Na overenie správnosti fungovania programu som použil príklady výsledkov z webstránky zadania. Program vypisuje aj postupnosť zo začiatočného stavu do koncového stavu. Po zistení správneho fungovania programu som porovnal obidve heuristiky na rovnakých vstupoch.

Heuristika č. 1 – Manhattanovská vzdialenosť				
Rozmer	Čas	Hĺbka	Sprac. uzlov	Gener. uzlov
3x2	0.007s	19	138	166
	0.009s	20	190	231
	0.009s	21	205	250
4x2	0.390s	34	6557	8370
	0.607s	35	11 242	13 434
	0.620s	36	11 531	13 729
3x3	0.124s	29	1796	2804
	0.257s	30	3731	5765
	0.222s	31	3481	5393
5x2	22.442s	50	316 196	415 286
	43.625s	53	691 882	860 471
	66.056s	55	1 058 889	1 245 453

Heuristika č. 2 – Hammingova vzdialenosť				
Rozmer	Čas	Hĺbka	Sprac. uzlov	Gener. uzlov
3x2	0.011s	19	249	290
	0.012s	20	280	313
	0.013s	21	297	324
4x2	0.908s	34	17 980	19 115
	0.967s	35	19 253	19 820
	1.017s	36	19 466	19 896
3x3	4.904s	29	79 903	104 277
	5.814s	30	96 504	121 438
	7.277s	31	120 870	144 468
5x2	98.011s	50	1 661 835	1 718 270
	110.318s	53	1 773 172	1 793 986
	110.161s	55	1 803 146	1 810 200



Po porovnaní oboch heuristík som prišiel k záveru, že heuristika manhattanovských vzdialeností je oveľa informovanešia a presnejšia, preto aj časy a počet generovaných uzlov je menší.

Ľubomír Dlhý, ID: 102897

Heuristika č. 1 – Manhattanovská vzdialenosť s rôznymi koeficientami prenásobenia									
Rozmer	Koeficient 1			Koeficient 2			Koeficient 10		
	Čas	Hĺbka	Spracov. uzlov	Čas	Hĺbka	Spracov. uzlov	Čas	Hĺbka	Spracov. uzlov
3x2	0.007s	19	138	0.006s	19	130	0.006s	19	127
	0.009s	20	190	0.006s	20	135	0.004s	20	85
	0.009s	21	205	0.007s	21	145	0.005s	21	100
4x2	0.390s	34	6557	0.113s	38	1677	0.031s	66	484
	0.607s	35	11 242	0.136s	35	2341	0.018s	65	299
	0.620s	36	11 531	0.163s	38	2712	0.019s	66	301
3x3	0.124s	29	1796	0.018s	33	259	0.015s	45	227
	0.257s	30	3731	0.015s	34	176	0.022s	44	299
	0.222s	31	3481	0.019s	33	242	0.009s	45	140
5x2	22.442s	50	316 196	0.718s	54	11 209	0.191s	114	2790
	43.625s	53	691 882	2.149s	59	34 014	0.104s	135	1339
	66.056s	55	1 058 889	4.455s	61	68 811	0.072s	139	979

Po prenásobení hodnoty heuristickej funkcie koeficientom sa výrazne zníži počet spracovaných uzlov a aj čas, avšak zväčší sa hĺbka – teda počet uzlov na ceste zo začiatočného stavu do koncového stavu.

```
1
 > Startovaci stav: ((5 6 2)(0 3 4)(1 8 7))
 > Koncovy stav:    ((7 8 3)(4 2 5)(0 6 1))
 > Nie je mozne najst riesenie pre tento hlavolam.
 > POCET SPRACOVANYCH UZLOV (KROKOV): 181440 VYGENEROVANYCH UZLOV: 181440
```

Nie všetky hlavolamy sa dajú vyriešiť, preto program ošetruje aj takýto stav. Po neúspešnom hľadaní cesty sa vypíše chybová hláška.

```
> Startovaci stav: ((5 11 6 3)(3 10 4 7)(7 2 9 1))
> Koncovy stav:    ((11 5 8 9)(9 1 6 10)(10 2 0 7))
> Vyprsal casovy alebo krokovy limit!
> POCET SPRACOVANYCH UZLOV (KROKOV): 266293 VYGENEROVANYCH UZLOV: 447771
```

V programe je nastavený časový limit 15 sekúnd a limit na počet krokov 10 miliónov. Po prekročení ktorejkoľvek z týchto hodnôt sa program zastaví a vypíše chybovú hlášku.

Ľubomír Dlhý, ID: 102897

ZHODNOTENIE

Zistil som, že je signifikantný rozdiel v efektivite algoritmu pri použití rôznych heuristických funkcií. Heuristika č. 1 – manhattanovská vzdialenosť bola omnoho efektívnejšia najmä pri zložitejších vstupoch, kde bolo potrebné generovať veľké množstvo uzlov v porovnaní s heuristikou č. 2 – hammingova vzdialenosť (počet zle položených políčok).

Zadanie som sa snažil riešiť efektívne, aj preto som zvolil jazyk C++, kde som využíval primitívne typy a vhodné dátové štruktúry. Program umožňuje vstup pre hlavolam ľubovoľného rozmeru. Na jeden uzol som používal m*n+7 bajtov pamäte. Na zamedzenie opakovaného spracovávania rovnakých stavov som využil hash tabuľku, ktorá zabráni zacykleniu a je z hľadiska časovej náročnosti efektívna. Uzly som vkladal do prioritného radu, ktorý vždy dá navrch uzol s najnižšou hodnotou. Ďalšou výhodou mojej implementácie je počítanie heuristických funkcií, ktoré sa pre všetky políčka hlavolamu počítajú iba raz – pri začiatočnom uzle, pri ostatných uzloch sa počítajú iba zo zmeny (posunutia 2 políčok).

Vylepšením mojej implementácie by bola lepšia heuristická funkcia (je použitá klasická manhattanova vzdialenosť a hammingova vzdialenosť), lepšia hashovacia funkcia pre hashovanie stavov (zatiaľ sa hashujú iba reťazce - stringy). Zefektívniť by sa dalo ukladanie stavov do uzlov, na ktoré je zatiaľ potreba m*n bajtov pamäte.

Program je závislý na jazyku C++ a na knižniciach:

- `iostream` - načítanie vstupu a výstupu z konzole
- `ctime` - počítanie času potrebného na vykonanie programu
- `string` - práca s reťazcami
- `vector` - vkladanie vrcholov do prioritného radu
- `queue` - prioritný rad (halda)
- `unordered_set` - hashová tabuľka
- `stdlib.h`