Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológii

Ilkovičova 2, 842 16 Bratislava 4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Zadanie č. 2, (8-hlavolam)

Dokumentácia

Predmet: Umelá Inteligencia

Autor: Ivan Gulis

Študijný program: Informatika

Semester: 4.

Ak. rok: 2014/2015

## Definovanie problému 2

*„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.“*

*Operátory sú len štyri:*

*VPRAVO, DOLE, VLAVO a HORE*

*„Operátor má jednoduchú úlohu - dostane nejaký stav a ak je to možné, vráti nový stav. Ak operátor na vstupný stav nie je možné použiť, výstup nie je definovaný. V konkrétnej implementácii je potrebné výstup buď vhodne dodefinovať, alebo zabrániť volaniu nepoužiteľného operátora.****Všetky operátory pre tento problém majú rovnakú váhu.“***

d) Použite algoritmus lačného hľadania, porovnajte výsledky heuristík 1. a 2.

 1. Počet políčok, ktoré nie sú na svojom mieste

 2. Súčet vzdialeností jednotlivých políčok od ich cieľovej pozície

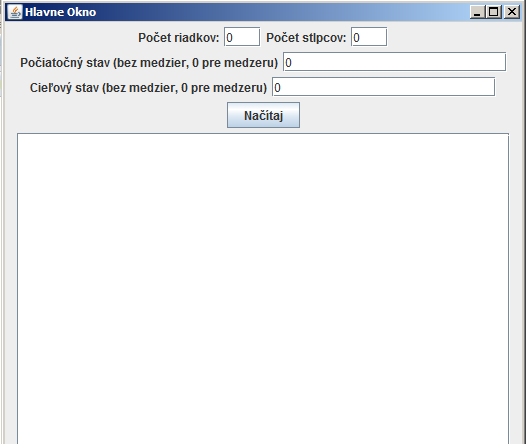
## Riešenie:

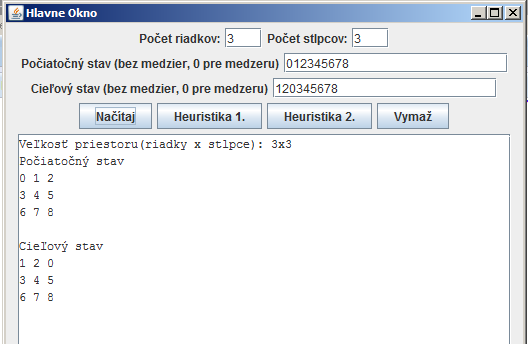
**Gui (používateľské rozhranie):**

Najskôr menší návod: Načítavanie vstupu riešim pomocou jednoduchého Gui. Skladá sa z tlačidla Načítaj, a textových polí, do ktorých je potrebné zadať rozmery hlavolamu, a počiatočný a koncový stav. Medzera musí byť vyjadrená číslom 0.

**NxM**

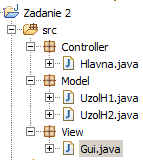
Je možné načítavať aj hlavolamy väčšie ako 3x3, ale dvojciferné čísla treba nahradiť písmenami (vstup na príklad 3, 4, ab0cdefghij8, ba0cdfeghij8). Program pracuje s charmi, takže je možná rôzna substitúcia znakov, ale 0 musí byť vždy medzera. Na konci dokumentácie Príklad 2, pre hlavolam 5x3 s využitím aj malých písmen abecedy.





Po stlačení tlačidla Načítaj sa objavia ďalšie tlačidlá, ktorými sa spúšťa hľadanie pre konkrétnu heuristiku. Pre nesprávny vstup (napr. rozmery sa nerovnajú počte vstupných znakov) program vypíše do TextArea chybovú hlášku o nekorektnom vstupe.

Pre implementáciu som zvolil prostredie Eclipse s jazykom Java, a návrhový vzor MVC(model, view, controller). Ku prepojeniu Gui a Controllera využívam vzor Singleton(jedináčik), takže controller obsahuje len jedinú inštanciu hlavnej triedy.



Hlavná trieda obsahuje väčšinu funkcii, ktoré sa behom programu volajú.

**Funkcia načítaj(),** ktorá inicializuje úvodné globáne premenné a zadefinuje počiatočný a koncový stav. Okrem toho prevedie vstupné stringy do polí, a zapamätá si pozíciu medzery.

**Funkcie naString() a naPole()** prevádzajú string na pole a naopak v jednoduchých cykloch.

Ďalej sú tam funkcie pre výpočet hodnôt heuristík sucetOdCiela() a pocetZlych().

Aplikácia lačného hľadania je v dvoch veľmi podobných funkciách heuristika1() a heuristika2(), kde heuristika1 pracuje s uzlami UzolH1 a heuristika2 pracuje s uzlami UzolH2. Uzly obsahujú atribúty o konkrétnom stave hlavolamu a predchodcovi.

## Opis algoritmu:

Jedná sa o lačné hľadanie, algoritmus si vždy „odhryzne“ ten najväčší kus. Je to druh hľadania „Najskôr najlepší“, a zameriava sa na odhad vzdialenosti ku cieľu. Využíva k tomu heuristiky „počet zle umiestnených políčok“ a „súčet vzdialeností políčok od cieľových pozícii“. Stále sú to ale odhady. Algoritmus vždy vyberie najlepší, aby sa dostal k cieľu čo najbližšie, to ale nemusí byť vždy najlepšie riešenie. **Algoritmus lačného hľadania nie je najoptimálnejší (nenájde najkratšiu cestu).**

Pre obe funkcie je postup podobný:

Najskôr sa inicializuje prvý startovací uzol. Načítané hodnoty sa uložia do premenných prvého objektu typu UzolH1 alebo UzolH2 (podľa funkcie), vytvorí sa nový HashSet a prioritný front. Do prioritného frontu sa vkladajú objekty uzlov. Obe triedy uzlov sú typu Comparable, a majú zadefinované compareTo funkcie, podľa reprezentujúcich heuristík, aby s nimi vedel prioritný front pracovať.

PriorityQueue<UzolH2> que = new PriorityQueue<UzolH2>();

Všetky vygenerované stavy ukladám do štruktúry HashSet, ktorá umožňuje rýchlu kontrolu, či stav už bol kedysi vygenerovaný.

HashSet<String> hashTable = new HashSet<String>();

Ukladám tam stavy v podobe stringov.

Po inicializácii prvého uzla ho pridám do radu, a spúšťam while cyklus:

1. Kým nie je front prázdny, vyber minimum
2. Skontroluj, či to nie je cieľ (ak je, skončí a nastaví premennú o nájdení cieľa na 1)
3. Pokús sa vygenerovať nasledovníkov (poradie: posun kocky dole, hore, doľava, doprava)
4. Vždy skontroluj HashSet, ak sa tam string nenachádza, pridaj ho do HashSetu a uzol do frontu medzi čakajúce uzly na spracovanie

Ak cyklus nájde cieľ, nastaví kontrolnú premennú na 1, a do StringBuilder premennej sa začne ukladať výpis.

## Konštrukcia cesty

Pri konštrukcii cesty využívam štruktúru ArrayList. Z posledného cieľového uzla získam smerník na rodiča, a vo while cykle sa dostanem až k počiatku. Cestou si ukladám stringy operátorov, ktoré následne odzadu pridám do výpisu. Vo výpise sú ešte informácie ako napr.:

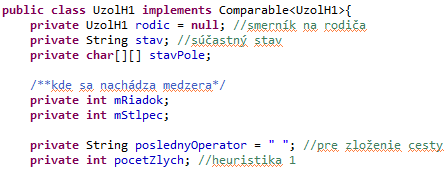
Dĺžka cesty: 118 (počet krokov)

Stavov prejdených: 910 (koľko stavov bolo vybraných ako aktuálny stav, **vrátane prvého**)

Stavov vygenerovaných: 1803 (koľko stavov sa vygenerovalo dostalo do frontu, **bez prvého**)

Čas: 12 ms (čas zbehnutia hľadania)

## Uzly



Každý uzol obsahuje smerník na rodiča(predchodcu), aby sa dal následne vystopovať koreň (ten má ako predchodcu nul).

V atribútoch stav a stavPole je uložený string a pole charov stavu hlavolamu.

Ďalej si každý stav pamätá pozíciu prázdneho miesta v mRiadok a mStlpec. Ušetrí to hľadanie.

Dôležitým atribútom pre konštrukciu cesty je operátor, ktorým bol stav vytvorený.

Každý uzol má potom atribút s hodnotou svojej heuristiky.

## Testovanie

Program som testoval cez Gui, načítavanie rôznych kombinácii začiatku a cieľa, korektných aj nekorektných vstupov. Pri konštrukcii stavov a podmienok som používal výpisy aktuálnych polí stavov do konzoly s informáciami najmä o správnom posune medzery. Pre zložité vstupy s veľkou dĺžkou cesty to už ale nebolo možné takto kontrolovať. Testovacie príklady sú na konci dokumentácie.

## Zhodnotenie riešenia

Implementovaný je funkčný algoritmus pre hľadanie riešenia 8-hlavolamu, ale aj hlavolamov iných rozmerov na podobný štýl problému.

Nevýhodou programu je nemožnosť načítania dvojciferných čísel, naopak dokáže pracovať s inými znakmi (ako napr. malé či veľké písmená), čo môže byť výhodou a spestrením. Nevýhodou môjho riešenia je existencia dvoch funkcii, jednej pre každú heuristiku, kde tieto funkcie sú skoro rovnaké. Problém spočíva vo funkcii prioritného frontu, kde musí byť jasne definovaná funkcia compareTo, ale pre 2 rôzne heuristiky je potrebné porovnávať 2 rôzne hodnoty. Oba problémy nie sú závislé od implementačného prostredia, len od samotnej implementácie.

Rozšírenie pre dvojciferné čísla je možné implementovať so zmenou načítavania, najjednoduchšie priamo z funkcie main respektíve z konzoly. Ďalšou možnosťou je zmeniť načítavaný vstup, a cifry rozdeľovať čiarkou alebo medzerou, a následne upraviť cyklus vo funkcii.

Čo sa implementačného prostredia týka, Eclipse poskytuje jednoduché a praktické prostredie a spolu s jazykom Java veľa predrobených dátových štruktúr, ktoré som v programe využíval (HashSet, ArrayList, PriorityQueue, StringBuilder...).

Práca s objektmi celú implementáciu podstatne zjednodušuje. Každý uzol je samostatný objekt, s ktorým sa dá pracovať prostredníctvom implementovaných funkcii.

## Vlastnosti metód

Obe heuristiky poskytujú inú dĺžku riešení, ani jedna nie je vždy lepšia ako druhá. V prvom príklade si vedú obe heuristiky rovnako, stavový priestor je veľmi malý.

V 2. príklade v veľkým stavovým priestorom vedie 2. heuristika, pretože má skoro 2x lepšie výsledky.

V 3. príklade naopak mierne vedie 1. heuristika. Povedal by som, že častejšie je úspešnejšia druhá heuristika, pretože jej samotné hodnoty sú presnejšie a informatívnejšie (odhad do cieľa je lepší). Kde prvá heuristika iba vracia počet zle umiestnených políčok, druhá heuristika okrem toho vracia aj vzdialenosť od cieľa pre každé políčko.

Výmena start-cieľ: Výsledky sú odlišné kvôli inej priorite vkladania uzlov do prioritného frontu. Najskôr sa pokúša vytvoriť nasledovníka pre pohyb kocky dole, hore, doľava, doprava. Pre cestu druhým smerom operátory nie sú rovnaké, takže sa môžu vygenerovať skôr iné stavy, a iné sa do frontu pre nájdenú duplicitu v HashSete ani nedostanú.

## Príklady

Na úvod príklad bez riešenia (while cyklus skončil neúspešne, obe funkcie nedokázali nájsť riešenie):

Veľkosť priestoru(riadky x stĺpce): 2x2

Počiatočný stav

1 2

3 0

Cieľový stav

1 3

2 0

Nepodarilo sa nájsť riešenie

Nepodarilo sa nájsť riešenie

**1. príklad**

Vstupy: 2, 3, 152403, 123450

Výstup:

Veľkosť priestoru(riadky x stĺpce): 2x3

Počiatočný stav

1 5 2

4 0 3

Cieľový stav

1 2 3

4 5 0

Heuristika 1.

Kroky:

DOLE VLAVO HORE

Dĺžka cesty: 3

Stavov prejdených: 4

Stavov vygenerovaných: 6

Čas: 1 ms

Heuristika 2.

Kroky:

DOLE VLAVO HORE

Dĺžka cesty: 3

Stavov prejdených: 4

Stavov vygenerovaných: 6

Čas: 1 ms

(naopak)

Veľkosť priestoru(riadky x stĺpce): 2x3

Počiatočný stav

1 2 3

4 5 0

Cieľový stav

1 5 2

4 0 3

Heuristika 1.

Kroky:

DOLE VPRAVO HORE

Dĺžka cesty: 3

Stavov prejdených: 4

Stavov vygenerovaných: 5

Čas: 0 ms

Heuristika 2.

Kroky:

DOLE VPRAVO HORE

Dĺžka cesty: 3

Stavov prejdených: 4

Stavov vygenerovaných: 5

Čas: 1 ms

**2. príklad**

Vstupy: 5, 3, 1524036789abcde, 123456789abcde0

Veľkosť priestoru(riadky x stĺpce): 5x3

Počiatočný stav

1 5 2

4 0 3

6 7 8

9 a b

c d e

Cieľový stav

1 2 3

4 5 6

7 8 9

a b c

d e 0

Heuristika 1.

Kroky:

DOLE VLAVO HORE HORE HORE HORE VPRAVO VPRAVO DOLE VLAVO HORE VLAVO DOLE VPRAVO HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE VPRAVO DOLE DOLE VPRAVO HORE HORE VLAVO VLAVO DOLE DOLE VPRAVO VPRAVO HORE HORE VLAVO VLAVO DOLE DOLE VPRAVO VPRAVO HORE HORE VLAVO VLAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO VPRAVO DOLE VLAVO HORE VLAVO DOLE VPRAVO DOLE VPRAVO HORE HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE VPRAVO DOLE VLAVO DOLE VPRAVO HORE VLAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO HORE VPRAVO DOLE VLAVO DOLE DOLE VPRAVO HORE HORE HORE VLAVO DOLE DOLE VPRAVO DOLE VLAVO HORE HORE HORE

Dĺžka cesty: 118

Stavov prejdených: 910

Stavov vygenerovaných: 1803

Čas: 12 ms

Heuristika 2.

Kroky:

DOLE VLAVO HORE HORE HORE VPRAVO VPRAVO HORE VLAVO VLAVO DOLE VPRAVO VPRAVO HORE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO VLAVO DOLE DOLE VPRAVO HORE HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE DOLE VPRAVO HORE VPRAVO HORE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO DOLE VLAVO HORE HORE HORE

Dĺžka cesty: 66

Stavov prejdených: 322

Stavov vygenerovaných: 614

Čas: 4 ms

(Výmena start/cieľ)

Veľkosť priestoru(riadky x stĺpce): 5x3

Počiatočný stav

1 2 3

4 5 6

7 8 9

a b c

d e 0

Cieľový stav

1 5 2

4 0 3

6 7 8

9 a b

c d e

Heuristika 1.

Kroky:

VPRAVO VPRAVO DOLE DOLE VLAVO VLAVO HORE VPRAVO DOLE DOLE VLAVO DOLE VPRAVO HORE HORE VPRAVO HORE VLAVO DOLE VPRAVO DOLE VLAVO DOLE VPRAVO HORE HORE VLAVO DOLE DOLE VPRAVO HORE HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE HORE VLAVO DOLE VPRAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO HORE HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO HORE VLAVO DOLE VPRAVO VPRAVO HORE VLAVO VLAVO DOLE VPRAVO VPRAVO HORE VLAVO DOLE VPRAVO HORE VLAVO DOLE VLAVO DOLE VPRAVO DOLE VLAVO HORE VPRAVO DOLE VLAVO HORE HORE VPRAVO VPRAVO HORE VLAVO DOLE VLAVO HORE VPRAVO VPRAVO DOLE VLAVO HORE VLAVO DOLE DOLE VPRAVO HORE VPRAVO DOLE VLAVO VLAVO HORE VPRAVO DOLE VPRAVO HORE HORE VLAVO DOLE VLAVO HORE VPRAVO VPRAVO DOLE VLAVO HORE VLAVO DOLE VPRAVO VPRAVO DOLE VLAVO HORE HORE VPRAVO DOLE VLAVO DOLE VPRAVO HORE HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE DOLE VPRAVO HORE HORE VLAVO HORE VPRAVO DOLE DOLE DOLE VLAVO HORE HORE HORE VPRAVO DOLE DOLE VLAVO DOLE VPRAVO HORE HORE HORE VLAVO DOLE DOLE VPRAVO DOLE VLAVO HORE HORE HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE DOLE VPRAVO DOLE VLAVO DOLE VLAVO HORE VPRAVO VPRAVO DOLE VLAVO HORE VLAVO DOLE VPRAVO VPRAVO HORE HORE VLAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO HORE

Dĺžka cesty: 230

Stavov prejdených: 6815

Stavov vygenerovaných: 13844

Čas: 43 ms

Heuristika 2.

Kroky:

DOLE DOLE DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO VPRAVO DOLE VLAVO VLAVO HORE VPRAVO DOLE VPRAVO HORE VLAVO VLAVO DOLE VPRAVO HORE HORE VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO DOLE VLAVO HORE HORE VPRAVO VPRAVO DOLE VLAVO HORE VLAVO DOLE DOLE VPRAVO VPRAVO HORE VLAVO VLAVO HORE HORE VPRAVO DOLE VLAVO DOLE VPRAVO VPRAVO DOLE VLAVO HORE VPRAVO HORE VLAVO HORE VPRAVO DOLE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO VLAVO HORE VPRAVO DOLE VPRAVO HORE VLAVO DOLE

Dĺžka cesty: 88

Stavov prejdených: 833

Stavov vygenerovaných: 1883

Čas: 15 ms

**3. príklad**

Vstupy: 3, 3, 012345678, 806547231

Veľkosť priestoru(riadky x stĺpce): 3x3

Počiatočný stav

0 1 2

3 4 5

6 7 8

Cieľový stav

8 0 6

5 4 7

2 3 1

Heuristika 1.

Kroky:

VLAVO VLAVO HORE HORE VPRAVO VPRAVO DOLE DOLE VLAVO VLAVO HORE HORE VPRAVO VPRAVO DOLE VLAVO DOLE VPRAVO HORE VLAVO DOLE VLAVO HORE VPRAVO DOLE VPRAVO HORE VLAVO DOLE VPRAVO HORE HORE VLAVO VLAVO DOLE DOLE VPRAVO HORE HORE VPRAVO DOLE VLAVO HORE VPRAVO DOLE VLAVO DOLE VLAVO HORE HORE VPRAVO VPRAVO DOLE VLAVO VLAVO DOLE VPRAVO HORE HORE VPRAVO DOLE VLAVO DOLE VPRAVO HORE VLAVO DOLE VLAVO HORE VPRAVO DOLE VPRAVO HORE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE VPRAVO HORE VLAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE DOLE

Dĺžka cesty: 105

Stavov prejdených: 414

Stavov vygenerovaných: 664

Čas: 1 ms

Heuristika 2.

Kroky:

HORE VLAVO DOLE VLAVO HORE VPRAVO DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO VLAVO HORE VPRAVO DOLE VPRAVO HORE VLAVO HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE VPRAVO DOLE DOLE VLAVO HORE HORE VPRAVO DOLE VLAVO DOLE VPRAVO HORE HORE VLAVO DOLE VPRAVO DOLE VPRAVO HORE VLAVO DOLE VPRAVO HORE VLAVO DOLE VLAVO HORE HORE VPRAVO DOLE VPRAVO HORE VLAVO VLAVO DOLE DOLE VPRAVO HORE VPRAVO HORE VLAVO DOLE DOLE VLAVO HORE HORE VPRAVO VPRAVO DOLE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO HORE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO VLAVO DOLE VPRAVO VPRAVO HORE HORE VLAVO DOLE VLAVO DOLE VPRAVO HORE HORE VPRAVO DOLE DOLE VLAVO VLAVO HORE VPRAVO DOLE VPRAVO HORE VLAVO VLAVO DOLE VPRAVO VPRAVO HORE HORE VLAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO

Dĺžka cesty: 135

Stavov prejdených: 519

Stavov vygenerovaných: 918

Čas: 2 ms

Výmena:

Veľkosť priestoru(riadky x stĺpce): 3x3

Počiatočný stav

8 0 6

5 4 7

2 3 1

Cieľový stav

0 1 2

3 4 5

6 7 8

Heuristika 1.

Kroky:

VPRAVO HORE HORE VLAVO VLAVO DOLE DOLE VPRAVO VPRAVO HORE HORE VLAVO VLAVO DOLE DOLE VPRAVO HORE VLAVO DOLE VPRAVO VPRAVO HORE VLAVO DOLE VLAVO HORE VPRAVO DOLE VLAVO HORE HORE VPRAVO VPRAVO DOLE DOLE VLAVO HORE HORE VLAVO DOLE VPRAVO HORE VLAVO DOLE VPRAVO DOLE VPRAVO HORE HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE VLAVO DOLE VPRAVO HORE VLAVO DOLE DOLE VPRAVO HORE HORE VLAVO DOLE DOLE VPRAVO HORE HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE VPRAVO DOLE VLAVO HORE HORE VPRAVO DOLE VPRAVO DOLE VLAVO HORE HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE VPRAVO DOLE VLAVO VLAVO HORE VPRAVO HORE VLAVO DOLE DOLE VPRAVO VPRAVO HORE VLAVO VLAVO HORE VPRAVO DOLE VPRAVO DOLE

Dĺžka cesty: 123

Stavov prejdených: 1023

Stavov vygenerovaných: 1645

Čas: 5 ms

Heuristika 2.

Kroky:

VPRAVO HORE HORE VLAVO DOLE DOLE VPRAVO HORE VLAVO DOLE VPRAVO HORE VLAVO VLAVO DOLE VPRAVO VPRAVO HORE VLAVO DOLE VPRAVO HORE HORE VLAVO DOLE DOLE VPRAVO HORE HORE VLAVO VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO DOLE VPRAVO HORE HORE VLAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE DOLE VLAVO HORE VPRAVO HORE VLAVO DOLE DOLE VPRAVO HORE VLAVO HORE VPRAVO DOLE VLAVO HORE VLAVO DOLE VPRAVO HORE VPRAVO DOLE VLAVO VLAVO HORE VPRAVO VPRAVO DOLE VLAVO HORE VLAVO DOLE VPRAVO VPRAVO HORE VLAVO DOLE VLAVO HORE VPRAVO VPRAVO DOLE DOLE

Dĺžka cesty: 93

Stavov prejdených: 446

Stavov vygenerovaných: 730

Čas: 2 ms

Iné vstupy na možné testovanie (výstupy neuvádzam, aby nebola dokumentácia príliš dlhá) :

Počiatočný stav:

243150786

Koncový stav:

123456780

Počiatočný stav:

412583a06b79

Koncový stav:

123456789ab0