



Semestrální práce z předmětu
UZI

Lloydova 15

19. listopadu 2024

Autor:

Miroslav Bártík

A22B0168P

`mbartik@students.zcu.cz`

1. Formulace úlohy

Ve čtverci $N \times N$ ($N \geq 3 \wedge N \leq 10$) je $N \cdot N - 1$ kostiček a jedno volné pole. Vytvořte program, který složí kostičky tak, aby byly uspořádané. Uspořádání vhodně zvolte. Program bude implementován jako konzolová aplikace. Aplikace bude také schopná generování hracího pole na základě zadaného počtu promíchávacích tahů. Uživatel bude mít také možnost hrací pole zadat sám pomocí výčtu jednotlivých řádek. Po vyřešení úlohy program vypíše jednotlivé kroky postupu, které vedly k vyřešení problému. Dále také vypíše statistiky, týkající se algoritmického řešení úlohy (čas hledání řešení, počet prohledaných stavů a počet nutných tahů).

2. Analýza úlohy

Možné přístupy k řešení

Existuje několik způsobů, jak úlohu 15 Puzzle řešit:

1. **Bruteforce přístup:** Vyzkoušet všechny možné sekvence tahů a vybrat tu, která vede k cílovému stavu. Tento přístup je však časově extrémně náročný a pro větší rozměry mřížky není praktický.
2. **Algoritmus BFS (Breadth-First Search):** Prohledávání do šířky hledá nejkratší cestu, ale je náročné na paměť, protože musí uchovávat všechny stavy na aktuální úrovni.
3. **Algoritmus DFS (Depth-First Search):** Prohledávání do hloubky se zaměřuje na jeden konkrétní směr hledání, ale nezaručuje nalezení nejkratší cesty.
4. **Algoritmus A* (zvolený přístup):** Kombinuje prohledávání do šířky s heuristickou funkcí, která odhaduje vzdálenost od cílového stavu. Tento přístup minimalizuje počet prozkoumaných stavů a zároveň zaručuje optimální řešení.

Výběr zvoleného přístupu

Byl zvolen algoritmus **A*** v kombinaci s **Manhattanovou heuristikou**. Tento přístup má následující výhody:

- Zaručuje nalezení optimálního řešení (nejkratšího počtu tahů).
- Heuristika (Manhattanova vzdálenost) efektivně odhaduje vzdálenost od cílového stavu, čímž redukuje počet prozkoumaných stavů.
- Je paměťově efektivnější než prohledávání do šířky.

Podmínky řešitelnosti

Počáteční stav je řešitelný, pokud splňuje následující podmínky:

- Pro liché N : Počet inverzí (párů dlaždic, kde větší číslo předchází menšímu) musí být sudý.
- Pro sudé N : Řešitelnost závisí na poloze prázdného pole:
 - Pokud je prázdné pole na sudé řádce (počítáno odspodu), počet inverzí musí být lichý.
 - Pokud je prázdné pole na liché řádce (počítáno odspodu), počet inverzí musí být sudý.

3. Popis algoritmu řešení

Formulace algoritmu A*

Algoritmus A* kombinuje hodnotu $f(n)$, která se skládá ze dvou složek:

$$f(n) = g(n) + h(n),$$

kde:

- $g(n)$: Počet tahů z počátečního stavu do aktuálního stavu.
- $h(n)$: Heuristická funkce odhadující zbývající vzdálenost k cílovému stavu (Manhattanova vzdálenost).

Manhattanova heuristika

Manhattanova vzdálenost je součet horizontálních a vertikálních vzdáleností každé dlaždice od její cílové pozice:

$$\text{Manhattanova vzdálenost} = \sum_{i=1}^N (|x_i - x_{i,\text{cílové}}| + |y_i - y_{i,\text{cílové}}|).$$

Je nutné podotknout, že tato heuristika je admisivní, což znamená, že nikdy nepřeceňuje skutečné náklady na přesunutí dlaždice.

Průběh algoritmu

1. Inicializace otevřené množiny (počáteční stav) a uzavřené množiny (prázdná).
2. Vybere se stav s nejnižší hodnotou $f(n)$ z otevřené množiny.
3. Pokud je stav cílový, algoritmus končí.
4. Jinak se stav přesune do uzavřené množiny a jeho sousední stavy se přidají do otevřené množiny. Pokračuje se krokem 2.

4. Popis programu (programová dokumentace)

Program je implementován v jazyce Python, což umožňuje snadné použití datových struktur (např. seznamů) a knihoven pro měření času a správu prioritních front.

Jako reprezentace samotné mřížky hlavolamu bylo zvoleno dvourozměrné pole, které programovací jazyk Python nabízí. Tato implementace byla zvolena zejména kvůli dobré představivosti, jelikož hlavolam má podobu mřížky dlaždic.

Významná proměnná, která se prolíná celým programem je `[n]`, což je velikost mřížky hlavolamu (počet dlaždic v ose X i v ose Y).

Struktura programu

Program je dělen dle standardní struktury znalostních systémů na následující moduly:

- **Com_module**: Modul, zajišťující komunikaci s uživatelem, skrze který je uživatel schopen dávat aplikaci input.
- **Data_base**: Modul, který uchovává data aplikace. V tomto případě je to samotný stav hry. Dále také nabízí metody pro přístup k podstatným informacím o datech a metody pro transformaci dat, které nesouvisí přímo se samotným algoritmickým řešením problému.
- **Explanation_module**: Modul, který zpracovává output programu do přehledné textové podoby a prezentuje jej uživateli.
- **Inference_module**: Standardní inferenční modul, který komunikuje s vedlejšími moduly a řídí chod aplikace.
- **K_base**: Báze znalostí, ve které je implementovaný samotný algoritmus (A^*) pro řešení úlohy, spolu s metodami pro testování existence řešení a správnosti řešení.
- **Main**: Vstupní bod programu pro uživatele.

5. Popis obsluhy programu (uživatelská dokumentace)

Pro spuštění aplikace je třeba mít instalovaný Python verze 3.12. Obsluha programu je popsána v následujících krocích:

1. Spusťte příkazový řádek v adresáři se všemi zdrojovými soubory.
2. Spusťte program pomocí příkazu `python Main.py`.
3. Zadejte aplikaci požadovanou velikost mřížky hlavolamu. Tato velikost musí být v rozmezí 3-10 včetně.

4. Program vám poskytne možnost mřížku ručně zadat, nebo náhodně vygenerovat.
 - V případě volby první možnosti bude program požadovat zadání mřížky po jednotlivých řádcích. Prázdnou dlaždici nahradíte symbolem 'X'
 - V případě druhé možnosti bude aplikace požadovat počet tahů, kterými bude složená mřížka zamíchána. Při pouhém stisknutí klávesy Enter bude použita výchozí hodnota (100).
5. Poté aplikace vypíše jednotlivé kroky, které vedly k vyřešení hlavolamu, spolu se statistikami o výsledném řešení (čas hledání řešení, počet prohledaných stavů a počet nutných tahů). Tato akce může trvat i vyšší jednotky minut v závislosti na velikosti a složitosti rozložení počáteční mřížky.
6. Aplikace dá poté uživateli možnost celý program opakovat.

6. Rozbor výsledků, zhodnocení

Aplikace dokáže pomocí algoritmu A* nalézt řešení problému Lloydova 15. Toto řešení je ideální (obsahuje nejmenší možný počet tahů) a je nalezeno vždy. Textové uživatelské rozhraní je přehledné a uživatelsky přívětivé (k tomu přispívají i barevné textové výpisy a ošetření všech uživatelských vstupů).

Ačkoliv pro jednoduché počáteční stavy program běží velice svižně, při složitějších výchozích stavech je hledání řešení velice pomalé. Přesné doby hledání řešení jsou zobrazeny v tabulce 1. Uvedené hodnoty představují průměrné výsledky z 5 běhů programu pro každou konkrétní výchozí konfiguraci. Pro získání těchto hodnot byl použit skript, který je součástí odevzdaného zip souboru (`Test.py`). Skript automaticky nastavuje výchozí parametry, spouští program a zaznamenává doby běhu i počet rozvinutých stavů.

Velikost mřížky (n)	míchací tahy	Čas (s)	Vrcholy
3	200	0.0810	1775
4	120	1.5619	19506
7	110	86.9261	276740
10	100	38.4769	75285

Tabulka 1: Výkonostní parametry pro různé velikosti mřížky a míchacích tahů v 5 iteracích.

Program je také pro složitější počáteční stav paměťově náročný, jelikož si algoritmus musí uchovávat velké množství stavů.

7. Závěr

Program splňuje požadavky zadání a umožňuje nalezení optimálního řešení Lloydova 15. Budoucí rozvoj může zahrnovat implementaci dalších heuristik nebo optimalizaci paměťové náročnosti. Dále je také možnost nahradit textové uživatelské rozhraní (TUI) grafický uživatelským rozhraním (GUI).