**Algoritmus DFS**

1. **Inicializácia**:  
   Algoritmus začína s počiatočným stavom hracej plochy 4x4 (prázdne pole a očíslované dlaždice) a pridáva ho do zásobníka na ďalšie preskúmanie. Prázdne pole predstavuje možné pohyby dlaždíc.
2. **Kontrola dokončenia**:  
   V každom kroku sa kontroluje, či aktuálne usporiadanie hracej plochy zodpovedá cieľovému stavu (dlaždice sú usporiadané od 1 po 15 a prázdne pole je v pravom dolnom rohu).
3. **Generovanie stavov**:  
   Z aktuálneho stavu sa generujú všetky možné pohyby prázdneho poľa (hore, dole, doľava, doprava, ak sú povolené). Nové stavy sa pridávajú do zásobníka, ak ešte neboli navštívené.
4. **Rekurzívne prehľadávanie**:  
   Algoritmus skúma každé možné riešenie do hĺbky, až kým nenájde cieľový stav alebo kým sa nevyčerpajú všetky možnosti. V prípade „slepej uličky“ (stav bez možných pohybov) sa algoritmus vráti o krok späť (backtracking).
5. **Vyhnutie sa cyklom**:  
   Na zabránenie opätovnému preskúmaniu rovnakých stavov sa používa dátová štruktúra (napríklad množina alebo hash tabuľka), ktorá uchováva už navštívené konfigurácie.
6. **Kontrola času a hĺbky**:  
   DFS môže byť pri veľkých problémoch časovo náročný. Preto je možné nastaviť maximálnu hĺbku prehľadávania a merať čas riešenia.
7. **Výsledok**:  
   Algoritmus vráti sekvenciu pohybov, ktoré vedú k riešeniu, alebo informáciu, že riešenie neexistuje.

DFS je efektívny pre malé problémy, ale pri veľkých priestoroch môže byť neefektívny, pretože prehľadáva všetky možnosti bez zaručenia optimálneho riešenia.

***Algoritmus A (Astar)*\***

1. **Inicializácia**:  
   Počiatočný stav sa pridá do prioritného radu (otvorený zoznam). Každý stav má hodnotu funkcie **f(n) = g(n) + h(n)**, kde:
   * **g(n)** — cena cesty od počiatočného stavu k aktuálnemu stavu.
   * **h(n)** — heuristická funkcia, ktorá odhaduje vzdialenosť od aktuálneho stavu k cieľovému stavu.
2. **Kontrola dokončenia**:  
   Algoritmus kontroluje, či je aktuálne usporiadanie dlaždíc totožné s cieľovým stavom.
3. **Generovanie stavov**:  
   Z aktuálneho stavu sa generujú všetky možné pohyby prázdneho poľa. Každý nový stav je ohodnotený funkciou **f(n)**.
4. **Prioritný rad**:  
   Stavy sa pridávajú do radu podľa hodnoty **f(n)**, pričom stavy s najnižšou hodnotou majú prioritu.
5. **Heuristická funkcia**:  
   Pre hru Pätnástky sa používa **Manhattan vzdialenosť**, ktorá spočíta súčet horizontálnych a vertikálnych vzdialeností všetkých dlaždíc od ich cieľových pozícií.
6. **Vyhnutie sa cyklom**:  
   Aby sa zabránilo opakovanému spracovaniu rovnakých stavov, používa sa „zatvorený“ zoznam, ktorý uchováva navštívené stavy.
7. **Výsledok**:  
   Ak algoritmus nájde cieľový stav, vráti optimálnu sekvenciu pohybov. Ak riešenie neexistuje, informuje o neúspechu.

A\* zaručuje nájdenie optimálneho riešenia, ak heuristika **h(n)** nie je nadhodnotená. Nevýhodou je vysoká spotreba pamäte pri veľkých problémoch.

### ****Algoritmus Greedy Search****

1. **Inicializácia**:  
   Počiatočný stav sa pridá do prioritného radu. Na rozdiel od A\*, Greedy Search používa iba heuristickú hodnotu **h(n)** bez zohľadnenia ceny cesty.
2. **Kontrola dokončenia**:  
   Algoritmus kontroluje, či je cieľový stav dosiahnutý.
3. **Generovanie stavov**:  
   Z aktuálneho stavu sa generujú všetky možné pohyby prázdneho poľa a hodnotí sa každý nový stav pomocou heuristiky **h(n)**.
4. **Prioritný rad**:  
   Stavy sa pridávajú do radu podľa hodnoty **h(n)**. Prednosť majú stavy s najnižšou heuristickou hodnotou.
5. **Heuristická funkcia**:  
   Podobne ako pri A\*, používa sa Manhattan vzdialenosť na odhad vzdialenosti od cieľového stavu.
6. **Vyhnutie sa cyklom**:  
   Algoritmus kontroluje, či aktuálny stav už nebol spracovaný, aby sa zabránilo cyklom.
7. **Výsledok**:  
   Algoritmus vráti nájdenú cestu k riešeniu alebo informáciu o tom, že riešenie neexistuje.

Greedy Search je rýchlejší než DFS a A\*, pretože sa zameriava na stavy s najnižšou heuristickou hodnotou. Avšak **nezaručuje optimálne riešenie**, pretože ignoruje cenu cesty.

**Porovnanie algoritmov na ukážkovej mape 3x3(level 1)**

|  |  |  |  |
| --- | --- | --- | --- |
| Názov algoritmu | DFS | *A (Astar)*\* | **Greedy Search** |
| Čas vykonania | 0.1483 sekúnd | 0.09826 sekúnd | 0.00804 sekúnd |
| Počet iterácii | 10220 | 26 | 56 |

**Porovnanie algoritmov na ukážkovej mape 4x4(level 6)**

|  |  |  |  |
| --- | --- | --- | --- |
| Názov algoritmu | DFS | *A (Astar)*\* | **Greedy Search** |
| Čas vykonania | Slnko zhasne skôr | > 20 min | 0.14456 sekúnd |
| Počet iterácii | ∞ | **>300 mln** | 176 |