ULB Sibiu

Algoritmul A* în Problema 8-Puzzle

Andrei-Nicolae Călugțiu

Sursă aplicație - Git

INTELIGENȚĂ ARTIFICIALĂ | 2023

Sursă aplicație - Git

ULB Sibiu

Cuprins

Introducere 8 Puzzle

- Un N-Puzzle conține √(n+1) poziții într-un careu în care una din poziții este liberă pentru a putea face mutări de rearanjare;
- Scopul este de a ajunge din starea inițială în starea finală, ambele fiind furnizate de la început;
- În fiecare pas se poate face o singură mutare, într-una din cele 4 direcții: sus, jos, stînga, dreapta, spre o poziție liberă;

- exemplu:

poz. inițială poziția finală

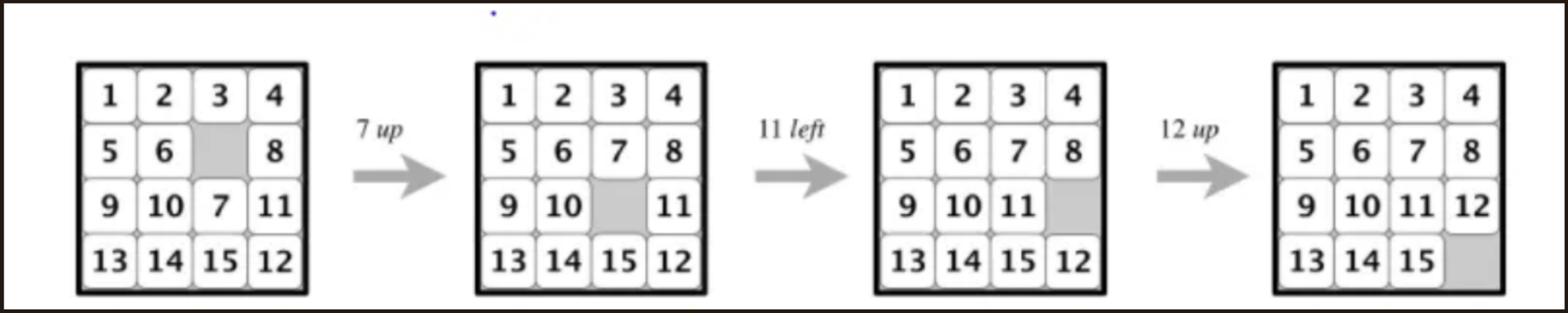
1	8	2		1	2	3
_	4	3	->	4	5	6
7	6	5		7	8	_

Sursă aplicație - Git

Verificarea unui careu valid

- Pentru ca un careu să fie rezolvabil, numărul de inversiuni trebuie să fie par;
- O inversiune este creată de o pereche de casete care sunt în ordinea inversă celei finale;

- Ideea este că paritatea invesiunilor rămîne egală după un set de mutări;
- Dacă ea este impară în situația de start, ea rămîne impară după oricîte mutări, iar dacă este pară în șituația de start, atunci după un număr finit de mutări, ea ajunge să fie 0, adică puzzle-ul este în starea finală cerută;

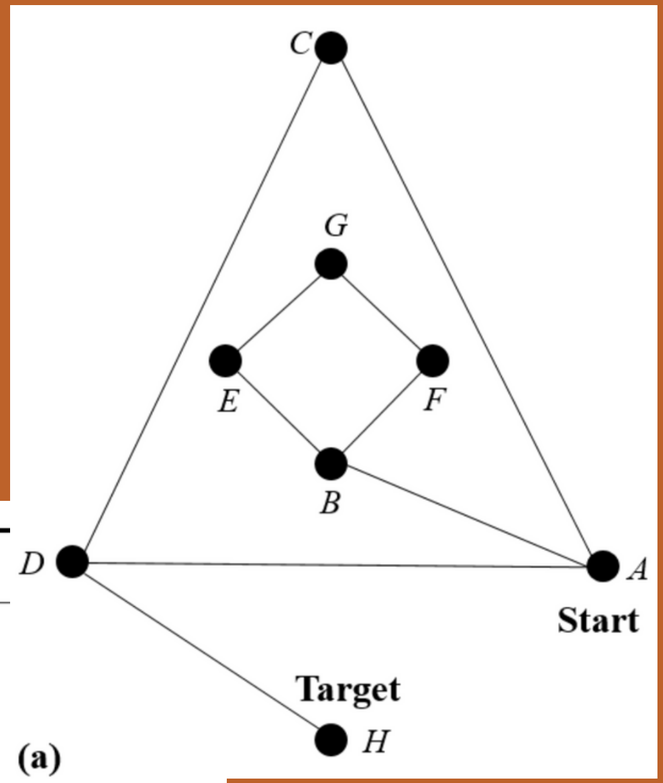


sursă: medium.com/Hà Huy Quân

Căutare euristică

- Un algoritm de căutare euristică știe unde sunt cele mai crescute șanse de a găsi elementul căutat și se îndreaptă spre acea zonă;
- O valoare generată de o funcție euristică, îi comunică algoritmului ce cale oferă cele mai mari șanse de a găsi elementul căutat. Această funcție se alege în funcție de problema de căutare;

sursă: Lancaster University



Step	Exploring	Open	Closed
1		$\{A\}$	$\{\}$
2	A	$\{B, C, D\}$	$\{A\}$
3	B	$\{C, D, E, F\}$	$\{A, B\}$
4	C	$\{D, E, F\}$	$\{A, B, C\}$
5	D	$\{E, F, H\}$	$\{A, B, C, D\}$
6	E	$\{F, H, G\}$	$\{A, B, C, D, E\}$
7	F	$\{H, G\}$	$\{A, B, C, D, E, F\}$
8	H	$\{G\}$	$\{A, B, C, D, E, F, H\}$
9	G	$\{\}$	$\{A, B, C, D, E, F, H, G\}$

(a)

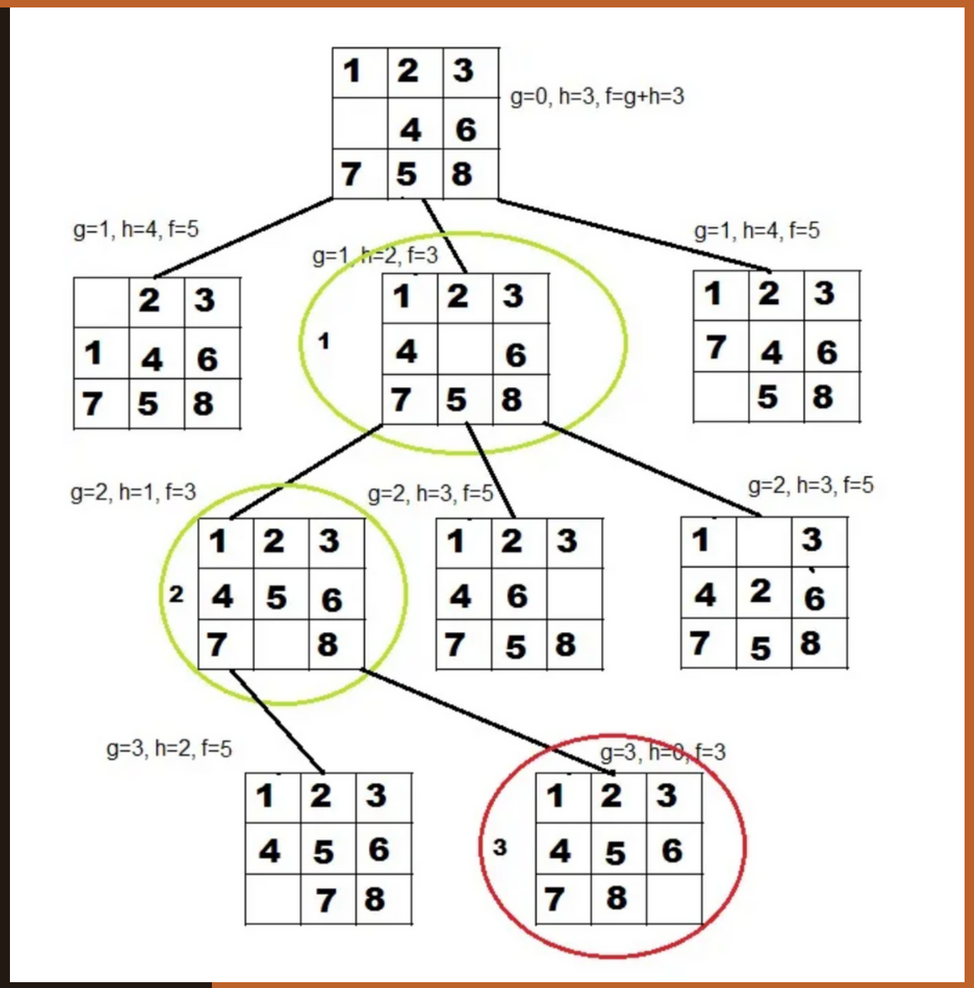
Algoritmul A*

- Utilizat larg pentru căutarea de rute și parcurgere de grafuri, acesta caută cea mai eficientă rută între mai multe noduri;
- Algoritmul ține o evidență cu toate nodurile vizitate și cu toate nodurile care mai sunt de vizitat, dar și o valoare de optimalitate pentru vizitarea fiecărui nod;
- Flecare nod reține adresa părintelui pentru a putea face ruta de întoarcere, iar expandarea nodurilor se face prin vizitarea și salvarea tuturor vecinilor nodului curent;
- Nodul optim de vizitat se alege după valoarea funcției **f = h + g**

- h -> distanța pînă la nodul țintă
- g -> numărul de noduri parcurse pînă la nodul curent

Sursă aplicație - Git

Aplicarea algoritmului la 8-Puzzle



sursă: medium.com/Ajinkya Sonawane

- Mutăm spațiul liber în toate direcțiile posibile și calculăm valoarea funcției f, în sensul în care expandăm starea inițială;
- Algoritmul alege nodul/starea cu cea mai mică valoare a funcției f, ceea ce înseamnă că aceasta are cea mai mare probabilitate de a ajunge mai repede la starea finală și expandează în continuare pînă atinge această stare;

Sursă aplicație - Git

ULB Sibiu

Sursă aplicație - Git

Mulțumesc!

Andrei-Nicolae Călușiu

INTELIGENȚĂ ARTIFICIALĂ | 2023