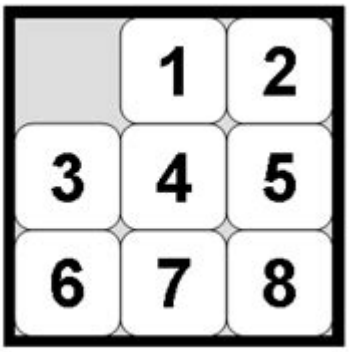
Laborator 2 și 3: Modelarea unei probleme de decizie

Problema: Avem o matrice 3x3 cu 8 dintre celule numerotate de la 1 la 8 și una goală. Știind că poziția inițială a celulelor este aleatoare și că putem muta o celulă doar în locul celulei goale și doar dacă este adiacentă acesteia, să se găsească, dacă există, o secvență de mutări ale celulelor astfel încât toate să fie plasate în ordine crescătoare în matrice. După mutarea unei celule, ea nu mai poate fi mutată din nou decât după ce unul din vecinii săi a fost mutat. Poziția celulei goale nu contează pentru validarea stării finale.



Folosiți pentru testarea implementărilor instanțele: [8, 6, 7, 2, 5, 4, 0, 3, 1], [2, 5, 3, 1, 0, 6, 4, 7, 8] și [2, 7, 5, 0, 8, 4, 3, 1, 6].

Etape de rezolvare:

1. (0.2) Alegeți o reprezentare a unei stări a problemei. Reprezentarea trebuie să fie suficient de explicită pentru a conține toate informaţiile necesare pentru continuarea găsirii unei soluții dar trebuie să fie și suficient de formalizată pentru a fi ușor de prelucrat/memorat.
2. (0.2) Identificați stările speciale (inițială și finală) și implementați funcția de inițializare (primește ca parametri instanța problemei, întoarce starea inițială) și funcția booleană care verifică dacă o stare primită ca parametru este finală.
3. (0.2) Implementați tranzițiile ca funcții care primesc parametri o stare și parametrii tranziției și întoarce starea rezultată în urma aplicării tranziției. Validarea tranzițiilor se face în una sau mai multe funcții booleană, cu aceeași parametri.
4. (0.4) Implementați strategia IDDFS.
5. (0.5) Implementați strategia Greedy și testați cel puțin trei euristici: distanța Manhattann, distanța Hamming, plus cel puțin încă o euristică diferită.
6. (0.5) Implementați un program care rulează toate cele 4 strategii (IDDFS și Greedy cu cele trei euristici) pentru cele trei instanțe și afișează soluția (dacă e găsită), lungimea sa (numărul de mutări) și durata execuției pentru fiecare din cele 4 strategii.
7. (Bonus: 0.1) Implementați strategia A* cu o euristică admisibilă. Includeți strategia în afișările de la punctul 6.

   Pentru rezolvarea incompletă a unei etape se acordă cel mult 0.1 puncte.

   Partea 1 (predare săptămâna 3): punctele 1-4.
   Partea 2 (predare săptămâna 4): punctele 5 și 6.

Resurse utile:

- https://deniz.co/8-puzzle-solver/
- https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/
- http://www.ieee.ma/uaesb/pdf/distances-in-classification.pdf

Link upload surse: https://forms.gle/Nx4BeyHE3KBoKACN6

Lab 2 and 3: Modeling a decision problem

Problem: We have a 3x3 matrix with 8 cells numbered 1 to 8 and one empty cell. Knowing that the initial position of the cells is random and that we can move a cell only in place of the empty cell and only if it is adjacent to it, find, if there is one, a sequence of moves such that all cells are placed in ascending order in the matrix. After a cell is moved it cannot be moved again until one of its neighbors has been moved. The position of the empty cell does not matter for final state validation.

Use these instances for testing: [8, 6, 7, 2, 5, 4, 0, 3, 1], [2, 5, 3, 1, 0, 6, 4, 7, 8] and [2, 7 , 5, 0, 8, 4, 3, 1, 6].

Steps to solve:

1. (0.2) Choose a representation of a state of the problem. The representation must be explicit enough to contain all the necessary information to continue finding a solution, but it must also be formal enough to be easy to process/store.
2. (0.2) Identify the special states (initial and final) and implement the initialization function (takes as parameters the problem instance, returns the initial state) and the boolean function that checks whether a state received as a parameter is final.
3. (0.2) Implement transitions as functions that take a state and transition parameters and return the state resulting from applying the transition. Validation of transitions is done in one or more boolean functions with the same parameters.
4. (0.4) Implement the IDDFS strategy.
5. (0.5) Implement the Greedy strategy and test at least three heuristics: Manhattann distance, Hamming distance, plus at least one other.
6. (0.5) Implement a program that runs all 4 strategies (IDDFS and Greedy with the three heuristics) for the three instances and displays the solution size (if found) and execution time for each of the 4 strategies.
7. (Bonus: 0.1) Implement the A* strategy with an admissible heuristic. Include the strategy in the displays in step 6.

   When displaying the solution, all the transitions made will be indicated, one by one.
   Part 1 (week 3): points 1-4.

Part 2 (Week 4): Points 5 and 6.

Useful resources:

- https://deniz.co/8-puzzle-solver/
- https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/
- http://www.ieee.ma/uaesb/pdf/distances-in-classification.pdf