```
SetOptions[$FrontEnd, DynamicEvaluationTimeout → Infinity];
(* Initial position of the tile x *)
initPosition[x_] := QuotientRemainder[x, 4] + {1, 1};

(* Heuristic function used to estimate how far is
 some state from the goal. More about it in README.md *)
h[state_] := Total[Table[(state[[i, j]] + 1) *
      ManhattanDistance[initPosition[state[[i, j]]], {i, j}], {i, 4}, {j, 4}], 2];

(* Function to determine where the space can be moved *)
possibleDirs[pos_] := {If[pos[[2]] > 1, {0, -1}, Nothing],
    If[pos[[2]] < 4, {0, 1}, Nothing],
    If[pos[[1]] > 1, {-1, 0}, Nothing],
    If[pos[[1]] < 4, {1, 0}, Nothing]
  };

(* Takes the current gameField and moves the
 tile from the position a in specified direction *)
move[gameField_, a_, dir_] := Block[{b, gf},
    b = a + dir;
    gf = gameField;
    {gf[[Sequence @@ a]], gf[[Sequence @@ b]]} = {gf[[Sequence @@ b]], gf[[Sequence @@ a]]};
    gf
  ];

DynamicModule[
 {sortedState, gameField, spacePos, solution, solve, displaySolution, shuffle},
 sortedState = Table[4 * i + j - 5, {i, 4}, {j, 4}];
 gameField = sortedState;
 spacePos = {1, 1};

 shuffle[] := Block[{moves, m},
   Do[
     moves = possibleDirs[spacePos];
     m = RandomChoice[moves];
     gameField = move[gameField, spacePos, m];
     spacePos += m
     , 2000];
  ];

 solve[] := Block[{startState, currentState, currentSpacePos, neighbors,
    state, newCost, ans, doRandomMoves, frontier, cameFrom, cost},
```

```
ans = {};
startState = gameField;

(* Does 50 random moves. It is useful in hard states and allows
 us to reach "average" state solvable by the main algorithm *)
doRandomMoves[] := Block[{moves, sp, prev},
  sp = FirstPosition[startState, 0];
  prev = Null;
  NestList[(
      prev = RandomChoice[possibleDirs[sp] // DeleteCases[-prev]];
      sp += prev;
      move[#, sp - prev, prev]) &, startState, 50]
 ];

While[True,
 frontier =
  CreateDataStructure["PriorityQueue", {{0, startState}}, First[#1] > First[#2] &];
 cameFrom = <|startState → Null|>;
 cost = <|startState → 0|>;

 (* A* algorithm with 10 sec time limit. When time limit reached,
 we do random moves and restart. *)
 TimeConstrained[
  While[True,
   currentState = Last@frontier["Pop"];
   If[currentState == sortedState, Break[], Null];
   currentSpacePos = FirstPosition[currentState, 0];
   neighbors =
    move[currentState, currentSpacePos, #] & /@ possibleDirs[currentSpacePos];

   newCost = cost[currentState] + 1;
   Scan[(
       If[newCost < Lookup[cost, Key[#], 1*^18],
         cost[#] = newCost;
         frontier["Push", {newCost + h[#], #}];
         cameFrom[#] = currentState,
         Null];
     ) &, neighbors];
  ];

  ans = Join[ans, Reverse@NestWhile[Append[#, cameFrom[Last@#]] &,
      {sortedState}, (cameFrom[Last@#] =!= Null) &]];
  Break[],
```

```
      10,
      ans = Join[ans, doRandomMoves[]];
      startState = Last@ans;
      ans = ans[[ ;; -2]];
     ];
    ];

    Return[ans];
  ];

 (* Animated solution *)
 displaySolution[] := Block[{},
    Scan[SessionSubmit[ScheduledTask[gameField = solution[[#]],
        {Quantity[# * 0.3, "Seconds"]}]] &, Range@Length[solution]];
    spacePos = {1, 1};
  ];

 (* Front-end *)
 EventHandler[Column[{
     Dynamic[Grid[gameField /. {0 → Null}, Frame → All,
       ItemSize → {3, 3}, ItemStyle → Directive[FontSize → 20]]],
     InputField[],
     Button["Shuffle", shuffle[]],
     Button["Solve", solution = solve[]; displaySolution[]]
    }],
  {"RightArrowKeyDown" :>
     If[spacePos[[2]] > 1, gameField = move[gameField, spacePos, {0, -1}];
      spacePos += {0, -1}, Null],
    "LeftArrowKeyDown" :>
     If[spacePos[[2]] < 4, gameField = move[gameField, spacePos, {0, 1}];
      spacePos += {0, 1}, Null],
    "DownArrowKeyDown" :>
     If[spacePos[[1]] > 1, gameField = move[gameField, spacePos, {-1, 0}];
      spacePos += {-1, 0}, Null],
    "UpArrowKeyDown" :> If[spacePos[[1]] < 4, gameField = move[gameField, spacePos, {1, 0}];
      spacePos += {1, 0}, Null]
  }]
]
```

Out[18]=

13	10	14	4
7	5	3	15
9	6		1
11	8	12	2

Shuffle

Solve

The tiles can be moved by arrows. To not lose the focus, you can use the input field. You can press any of the buttons at any time:

· Shuffle makes a large number of random movements starting from the current position.

· Solve searches for and displays the solution to the puzzle.