AI Assignment Report

Discussion of relative merits of search strategies

Advantages of BFS:-

1. Solution will definitely found out by BFS If there are some solution.

2. BFS will never get trapped in blind alley , means unwanted nodes.

3. If there are more than one solution then it will find solution with minimal steps.

Disadvantages Of BFS :-

1. Memory Constraints As it stores all the nodes of present level to go for next level.

2. If solution is far away then it consumes time.

Advantages Of DFS :-

1. Memory requirement is Linear WRT Nodes.

2. Less time and space complexity rather than BFS.

3. Solution can be found out by without much more search.

Disadvantage of DFS :-

1. Not Guaranteed that it will give you solution.

2. Cut-off depth is smaller so time complexity is more.

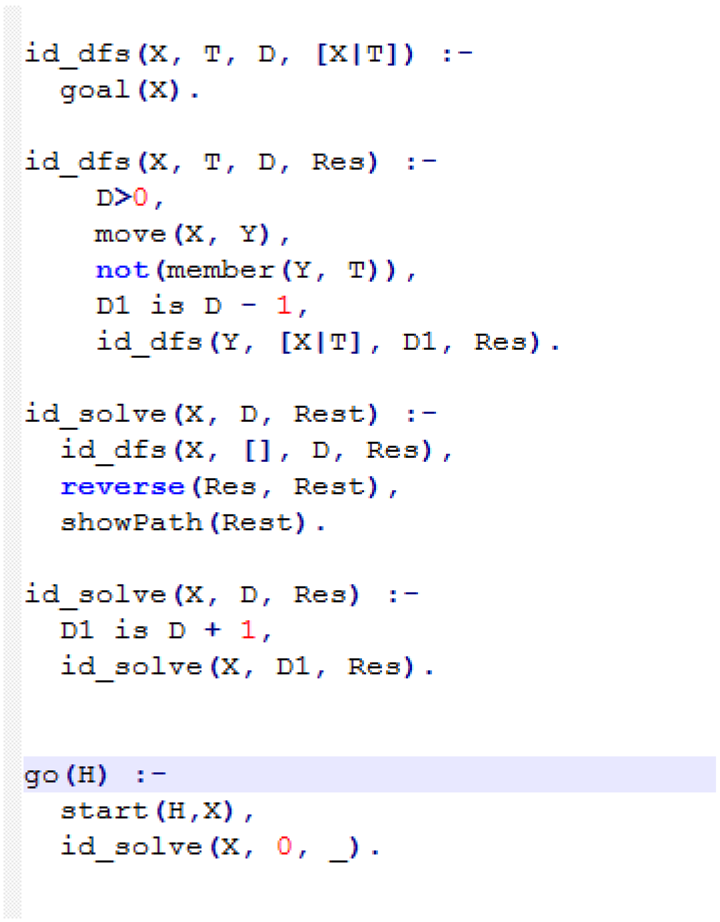
3. Determination of depth until the search has proceeds.

```
move( [E | Tiles] , [T| Tiles1] ):-
    swap( E , T , Tiles , Tiles1 ) .


swap( E , T , [T | Ts] , [E | Ts] ):-
    mandist( E , T , 1 ) .

swap( E , T , [T1 | Ts] , [T1 | Ts1] ):-
    swap( E , T , Ts , Ts1 ) .



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Manhattan Distance - mandist( TilePos1 , TilePos2, Di
%   is the distance between two tile positions .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mandist( (X,Y) , (X1,Y1) , D ):-
    diff( X , X1 , Dx ) ,
    diff( Y , Y1 , Dy ) ,
    D is Dx + Dy .

diff( A , B , D ):-
    D is A - B , D > 0 , !
    ;
    D is B - A .
```

this code is to find which cell can swap with null cell. D is distance between cell and null cell, if distance is 1 then they can swap if not they can not swap.

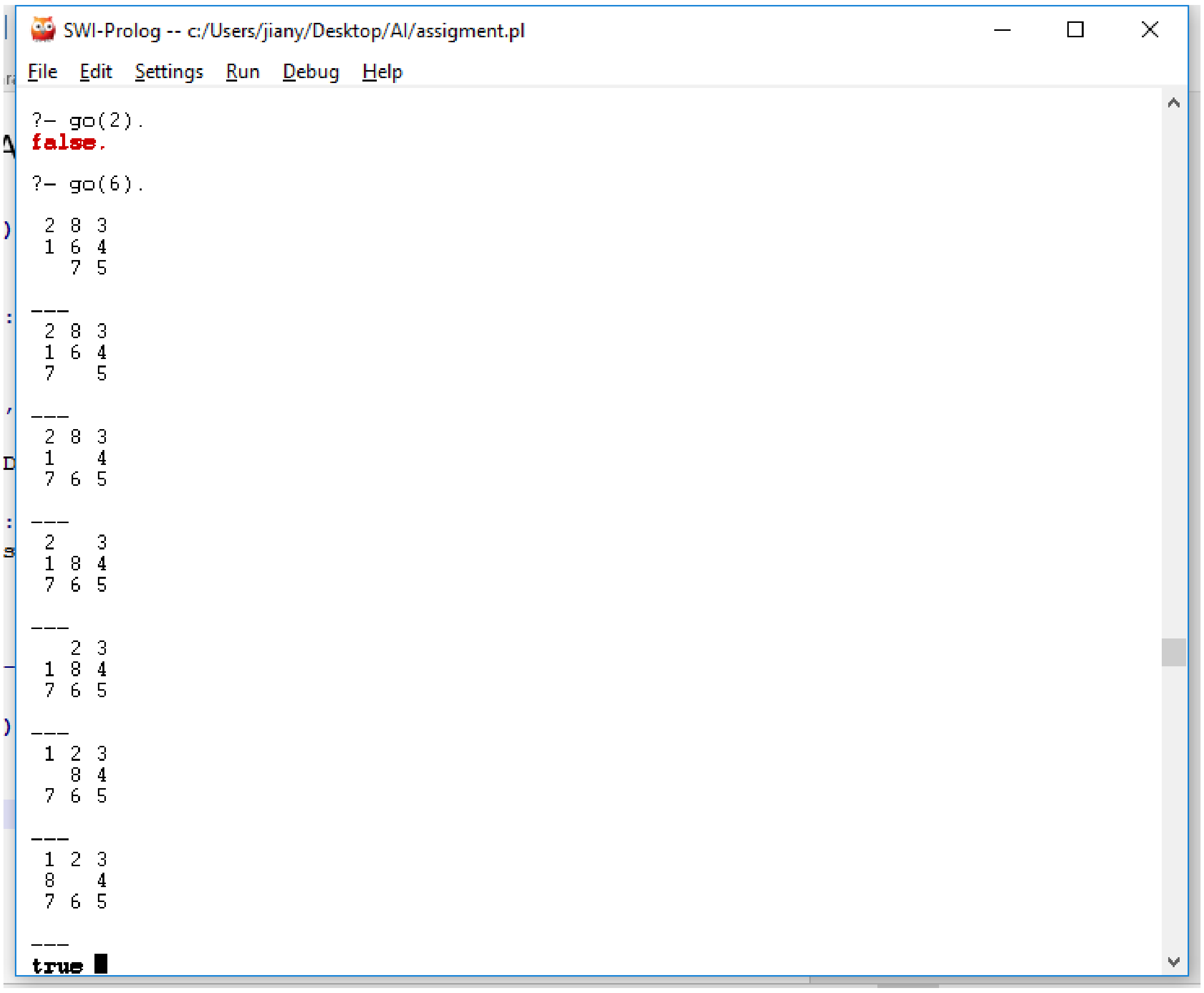
```
showPath( [] ) .
showPath( [P | L] ) :-
    showState( P ),
    nl, write('---'),
    showPath( L ).


showState([P0, P1, P2, P3, P4, P5, P6, P7, P8]) :-
    member( Y , [1, 2, 3] ),
    nl,
    member( X , [1, 2, 3] ),
    member( Tile-(X,Y),
            [' '-P0, 1-P1, 2-P2, 3-P3, 4-P4, 5-P5, 6-P6, 7-P7, 8-P8] ) ,
    write(' '), write( Tile ) ,
    fail
    ;
    nl, true .
```

This code is shows each step of swap of cells.



```
id_dfs(X, T, D, [X|T]) :-
  goal(X).

id_dfs(X, T, D, Res) :-
    D>0,
    move(X, Y),
    not(member(Y, T)),
    D1 is D - 1,
    id_dfs(Y, [X|T], D1, Res).

id_solve(X, D, Rest) :-
  id_dfs(X, [], D, Res),
  reverse(Res, Rest),
  showPath(Rest).

id_solve(X, D, Res) :-
  D1 is D + 1,
  id_solve(X, D1, Res).


go(H) :-
  start(H,X),
  id_solve(X, 0, _).
```

This code is to find solution of each depth, if no solution go next depth or go back last depth to find next, until find the goal.


SWI-Prolog -- c:/Users/jiany/Desktop/AI/assigment.pl
File Edit Settings Run Debug Help

```
?- go(2).
false.

?- go(6).

 2 8 3
 1 6 4
   7 5

---
 2 8 3
 1 6 4
 7   5

---
 2 8 3
 1   4
 7 6 5

---
 2   3
 1 8 4
 7 6 5

---
   2 3
 1 8 4
 7 6 5

---
 1 2 3
   8 4
 7 6 5

---
 1 2 3
 8   4
 7 6 5

---
true
```

After implement, the result shows.