E02 15 Puzzle Problem

Suixin Ou

School of Computer Science
Sun Yat-sen University

September 7, 2021

Task

Problem

Please solve 15-Puzzle problem by using IDA* (Python or C++). You can use one of the two commonly used heuristic functions: h1 = the number of misplaced tiles. h2 = the sum of the distances of the tiles from their goal positions.

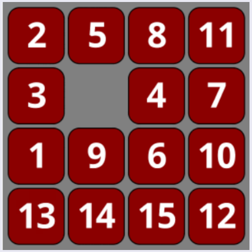


Figure 1: Searching by IDA*

Task

Input-output

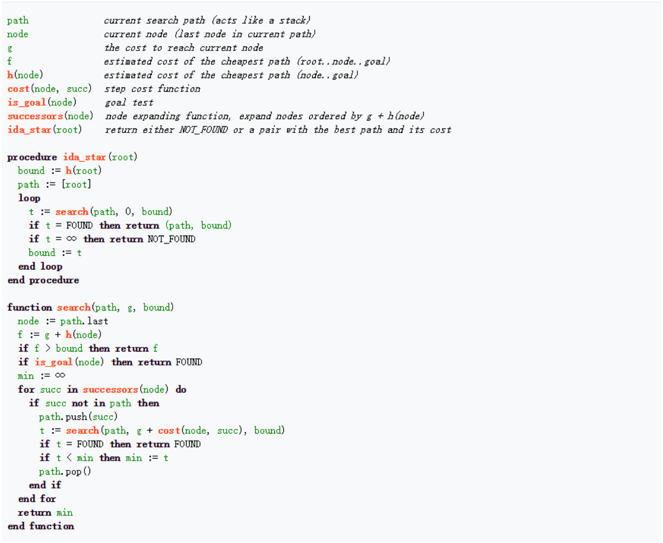
- Input: a 4x4 matrix of initial state.
- Output: the path from the initial state to the terminate state.

Submission

pack your report `E02_YourNumber.pdf` and source code into zip file `E02_YourNumber.zip`, then send it to `ai_course2021@163.com`.

Solution

Algorithm procedure



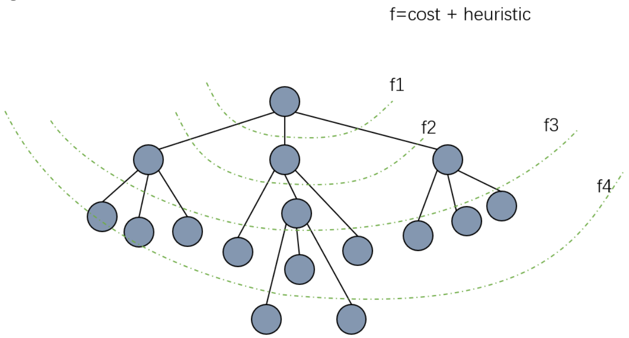
```
path              current search path (acts like a stack)
node              current node (last node in current path)
g                 the cost to reach current node
f                 estimated cost of the cheapest path (root..node..goal)
h(node)           estimated cost of the cheapest path (node..goal)
cost(node, succ)  step cost function
is_goal(node)     goal test
successors(node)  node expanding function, expand nodes ordered by g + h(node)
ida_star(root)    return either NOT_FOUND or a pair with the best path and its cost

procedure ida_star(root)
  bound := h(root)
  path := [root]
  loop
    t := search(path, 0, bound)
    if t = FOUND then return (path, bound)
    if t = ∞ then return NOT_FOUND
    bound := t
  end loop
end procedure

function search(path, g, bound)
  node := path.last
  f := g + h(node)
  if f > bound then return f
  if is_goal(node) then return FOUND
  min := ∞
  for succ in successors(node) do
    if succ not in path then
      path.push(succ)
      t := search(path, g + cost(node, succ), bound)
      if t = FOUND then return FOUND
      if t < min then min := t
      path.pop()
    end if
  end for
  return min
end function
```

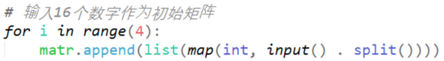
Solution

Algorithm illustration



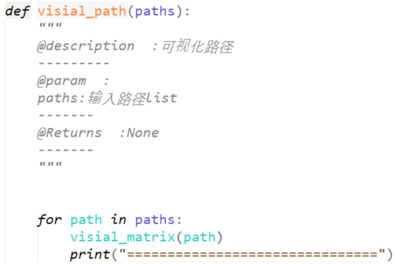
Solution

Read input



```python
# 输入16个数字作为初始矩阵
for i in range(4):
    matr.append(list(map(int, input() . split())))
```

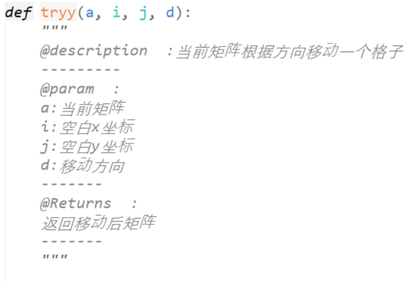
Visualize output



```python
def visial_path(paths):
    """
    @description  :可视化路径
    ---------
    @param  :
    paths:输入路径list
    -------
    @Returns  :None
    -------
    """


    for path in paths:
        visial_matrix(path)
        print("================================")
```

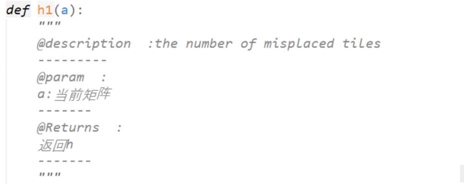
Get next state



```python
def tryy(a, i, j, d):
    """
    @description  :当前矩阵根据方向移动一个格子
    ---------
    @param  :
    a:当前矩阵
    i:空白x坐标
    j:空白y坐标
    d:移动方向
    -------
    @Returns  :
    返回移动后矩阵
    -------
    """
```

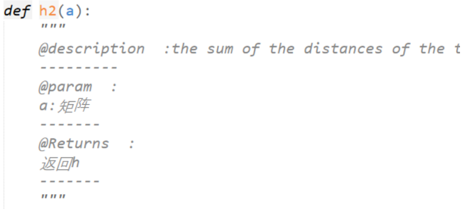
Solution

Heuristic 1



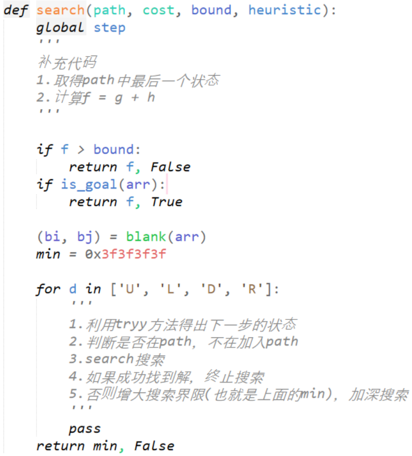
```
def h1(a):
    """
    @description  :the number of misplaced tiles
    ---------
    @param  :
    a:当前矩阵
    -------
    @Returns  :
    返回h
    -------
    """
```

Heuristic 2



```
def h2(a):
    """
    @description  :the sum of the distances of the t
    ---------
    @param  :
    a:矩阵
    -------
    @Returns  :
    返回h
    -------
    """
```

Search



```
def search(path, cost, bound, heuristic):
    global step
    '''
    补充代码
    1.取得path中最后一个状态
    2.计算f = g + h
    '''

    if f > bound:
        return f, False
    if is_goal(arr):
        return f, True

    (bi, bj) = blank(arr)
    min = 0x3f3f3f3f

    for d in ['U', 'L', 'D', 'R']:
        '''
        1.利用tryy方法得出下一步的状态
        2.判断是否在path, 不在加入path
        3.search搜索
        4.如果成功找到解, 终止搜索
        5.否则增大搜索界限(也就是上面的min), 加深搜索
        '''
        pass
    return min, False
```

The End