

# Lab2 15-Puzzle Problem (IDA\*)

---

17341015 Hongzheng Chen

September 7, 2019

## Contents

<b>1</b>	<b>IDA* Algorithm</b>	<b>2</b>
1.1	Description . . . . .	2
1.2	Pseudocode . . . . .	2
<b>2</b>	<b>Tasks</b>	<b>3</b>
<b>3</b>	<b>Codes</b>	<b>3</b>
<b>4</b>	<b>Results</b>	<b>6</b>

# 1 IDA\* Algorithm

## 1.1 Description

Iterative deepening A\* (IDA\*) was first described by Richard Korf in 1985, which is a graph traversal and path search algorithm that can find the shortest path between a designated start node and any member of a set of goal nodes in a weighted graph.

It is a variant of **iterative deepening depth-first search** that borrows the idea to use a heuristic function to evaluate the remaining cost to get to the goal from the **A\* search algorithm**.

Since it is a depth-first search algorithm, its memory usage is lower than in A\*, but unlike ordinary iterative deepening search, it concentrates on exploring the most promising nodes and thus does not go to the same depth everywhere in the search tree.

**Iterative-deepening-A\* works as follows:** at each iteration, perform a depth-first search, cutting off a branch when its total cost  $f(n) = g(n) + h(n)$  exceeds a given threshold. This threshold starts at the estimate of the cost at the initial state, and increases for each iteration of the algorithm. At each iteration, the threshold used for the next iteration is the minimum cost of all values that exceeded the current threshold.

## 1.2 Pseudocode





```
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
```

## 2 Tasks

- 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.
- Here are 4 test cases for you to verify your algorithm correctness. You can also play this game (15puzzle.zip) for more information.

	<p>TextOut of Result</p> <pre> 11 3 1 7 4 6 8 2 15 9 10 13 14 12 5 0 LowerBound 38 moves A optimal solution 58 moves Used time 3 sec 13 10 8 6 9 12 5 13 10 8 12 15 14 5 13 12 15 14 5 13 14 9 4 11 3 1 6 4 11 3 1 6 4 2 8 10 12 15 10 8 7 4 2 11 3 5 9 10 11 3 6 2 3 7 8 12 </pre>		<p>TextOut of Result</p> <pre> 14 10 6 0 4 9 1 8 2 3 5 11 12 13 7 15 LowerBound 37 moves A optimal solution 49 moves Used time 0 sec 6 10 9 4 14 9 4 1 10 4 1 3 2 14 9 1 3 2 5 11 8 6 4 3 2 5 13 12 14 13 12 7 11 12 7 14 13 9 5 10 6 8 12 7 10 6 7 11 15 </pre>
	<p>TextOut of Result</p> <pre> 0 5 15 14 7 9 6 13 1 2 12 10 8 11 4 3 LowerBound 44 moves A optimal solution 62 moves Used time 4 sec 7 9 2 1 9 2 5 7 2 5 1 11 8 9 5 1 6 12 10 3 4 8 11 10 12 13 3 4 8 12 13 15 14 3 4 8 12 13 15 14 7 2 1 5 10 11 13 15 14 7 3 4 8 12 15 14 11 10 9 13 14 15 </pre>		<p>TextOut of Result</p> <pre> 6 10 3 15 14 8 7 11 5 1 0 2 13 12 9 4 LowerBound 32 moves A optimal solution 48 moves Used time 0 sec 9 12 13 5 1 9 7 11 2 4 12 13 9 7 11 2 15 3 2 15 4 11 15 8 14 1 5 9 13 15 7 14 10 6 1 5 9 13 14 10 6 2 3 4 8 7 11 12 </pre>

- Please send E02\_YourNumber.pdf to ai\_201901@foxmail.com, you can certainly use E02\_15puzzle.tex as the L<sup>A</sup>T<sub>E</sub>X template.

## 3 Codes

In this lab, I implement the IDA\* algorithm in Python.

```

import sys
from copy import deepcopy
from queue import PriorityQueue

MAX_INT = 0x3f3f3f3f

def print_arr(arr):
    """
    Print arr (4*4)
    """
    if arr == None:
        print(arr)
    else:
        for i in range(4):
            print(*(arr[i]))
        print()

def get_target_pos(num):

```

```

"""
Get target position of num
"""
return ((num-1) // 4, (num-1) % 4) if num != 0 else (3,3)

def heuristics(a):
    """
    Heuristic function for IDA*
    """
    res = 0
    for i in range(4):
        for j in range(4):
            (ti, tj) = get_target_pos(a[i][j])
            res += abs(i - ti) + abs(j - tj)
    return res

def find_space(a):
    """
    Find the space of the matrix a
    """
    for i in range(4):
        try:
            j = a[i].index(0)
            return (i,j)
        except:
            pass

def move(a,i,j,d):
    """
    An action on matrix 'a' that moves (i,j)
    in the direction of 'd'
    """
    res = deepcopy(a) # be careful!
    if d == 'U':
        if i+1 < 4:
            res[i][j], res[i+1][j] = res[i+1][j], res[i][j]
            return res
        else:
            return None
    elif d == 'L':
        if j+1 < 4:
            res[i][j], res[i][j+1] = res[i][j+1], res[i][j]
            return res
        else:
            return None
    elif d == 'D':
        if i-1 >= 0:
            res[i][j], res[i-1][j] = res[i-1][j], res[i][j]
            return res
        else:
            return None
    elif d == 'R':
        if j-1 >= 0:
            res[i][j], res[i][j-1] = res[i][j-1], res[i][j]
            return res
        else:
            return None

```

```

        return None
    else:
        raise RuntimeError

def search(path, cost, bound):
    """
    Path: The previous states
    Cost: The current cost (f)
    Bound: Maximum limitation of IDA*
    """
    arr = path[-1]
    f = cost + heuristics(arr)
    if f > bound:
        return f, False
    if arr == [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,0]]:
        return f, True # h = 0
    (si,sj) = find_space(arr)

    # find the successors
    # based on the f+g priority
    q = PriorityQueue()
    for d in ['U','L','D','R']:
        new_state = move(arr,si,sj,d) # not share
        if new_state in path:
            continue
        if new_state != None:
            priority = cost + 1 + heuristics(new_state)
            q.put((priority,new_state))

    # DFS
    minF = MAX_INT
    while not q.empty():
        _, state = q.get()
        path = path + [state]
        f, found = search(path, cost+1, bound)
        if found == True:
            return f, True
        else:
            minF = min(f, minF)
        del path[-1]
    return minF, False

def ida_star(src):
    """
    Main function of IDA*
    """
    bound = heuristics(arr)
    path = [src]
    while True:
        bound, found = search(path, 0, bound)
        if found == True:
            break
        if bound == MAX_INT:
            print("NOT FOUND!")
            return
    print(bound)

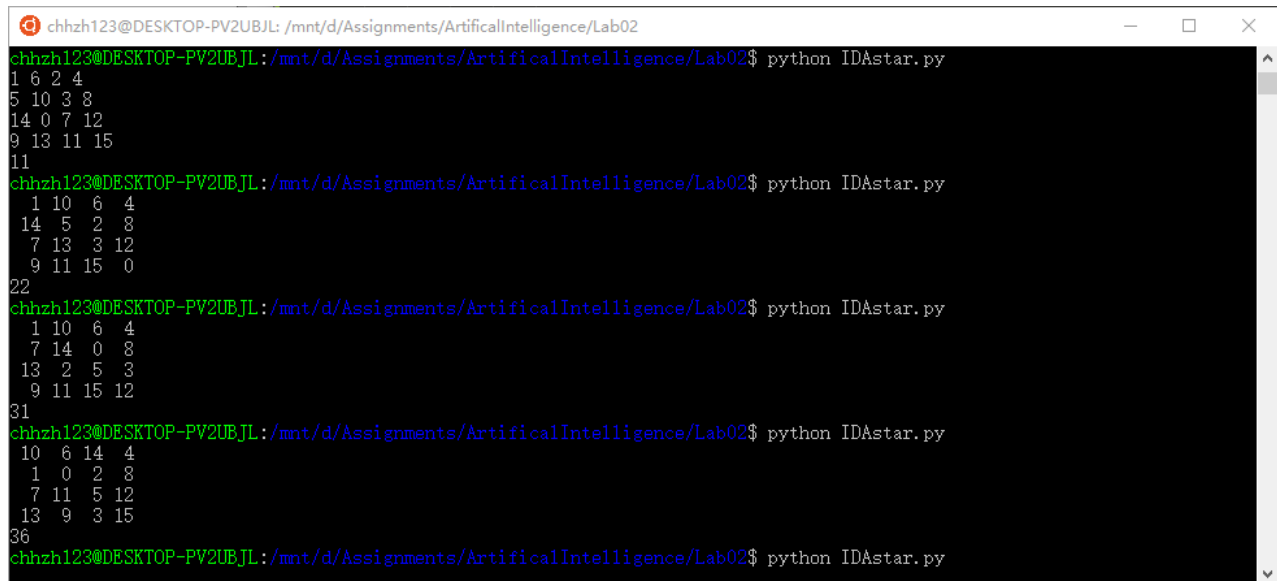
```

```
# Read in data
arr = []
for i in range(4):
    arr.append(list(map(int,input().split())))

# Initialization
ida_star(arr)
```

## 4 Results

The results are shown below. Due to the selection of the heuristic function and the implementation language, my code cannot scale well, thus only problems within 40 steps can be solved in 1 minute.



```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ArtificialIntelligence/Lab02
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/Lab02$ python IDAstar.py
1 6 2 4
5 10 3 8
14 0 7 12
9 13 11 15
11
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/Lab02$ python IDAstar.py
1 10 6 4
14 5 2 8
7 13 3 12
9 11 15 0
22
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/Lab02$ python IDAstar.py
1 10 6 4
7 14 0 8
13 2 5 3
9 11 15 12
31
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/Lab02$ python IDAstar.py
10 6 14 4
1 0 2 8
7 11 5 12
13 9 3 15
36
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/Lab02$ python IDAstar.py
```