

## 1. 硬體規格

---

- 筆電
  - CPU : Intel(R) Core(TM) i7-10510U CPU @ 1.60GHz (8核)
  - 顯卡 : NVIDIA GeForce MX250
  - RAM : 12GB
- 桌電
  - CPU : Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz (8核)
  - 顯卡 : NVIDIA GeForce GTX 1060 3GB
  - RAM : 16GB

桌電大概七年前買的，顯卡後來有更新過，筆電是兩年前買的。作業系統皆為 Windows10，沒有其他規格的電腦可以選了，如果筆電顯卡跑不動實驗，會讓桌機去跑。

電話：0901228989

## 2. 輸入檔

---

easy\_input.txt

```
1 | 5 4
2 | 1 2 2 3
3 | 1 2 2 3
4 | 5 5 0 6
5 | 4 8 0 6
6 | 4 7 910
```

medium\_input.txt

```
1 | 5 4
2 | 0 1 2 2
3 | 0 1 2 2
4 | 5 5 6 3
5 | 4 8 6 3
6 | 4 7 910
```

hard\_input.txt

```
1 | 5 4
2 | 1 2 2 0
3 | 1 2 2 3
4 | 5 5 8 3
5 | 4 9 0 6
6 | 4 710 6
```

no\_answer\_input.txt

```
1 5 4
2 3 2 2 1
3 3 2 2 1
4 4 5 5 6
5 4 0 0 6
6 7 8 9 10
```

製作方式皆為隨便移動步數，越少的越簡單

### 3. IDS

使用方式：

1. terminal 輸入 `python3 ids.py` 後會跳出以下訊息

```
Please enter your input file name(ex. input.txt):
```

2. 按照要求輸入輸入檔名稱

```
Please enter your input file name(ex. input.txt): easy_input.txt
```

3. 輸出結果，結果會自動存取於 `output.txt`

```
λ python3 ids.py

Please enter your input file name(ex. input.txt): easy_input.txt
complete reading input, use time: 0.0 seconds

starting ids, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDS

Total run time = 0.004985809326171875 seconds.
An optimal solution has 4 moves:
5R 4U 7L 8D
```

source code:

```
1 import time
2
3 num_map = []
4 ans_map = [[1, 2, 2, 3], [1, 2, 2, 3], [4, 5, 5, 6], [4, 0, 0, 6], [7, 8,
5 9, 10]]
6 # decide max depth for IDS
7 max_depth = 12
8
9 height = 0
10 width = 0
11
12 # check if the move is valid
13 def checkMove(num_map, num, dir):
14     for y in range(height):
15         for x in range(width):
```

```

15         if dir == 'R':
16             if num_map[y][x] == num and x == width - 1:
17                 return False
18             if num_map[y][x] == num and (num_map[y][x + 1] != num and
num_map[y][x + 1] != 0):
19                 return False
20         elif dir == 'L':
21             if num_map[y][x] == num and x == 0:
22                 return False
23             if num_map[y][x] == num and (num_map[y][x - 1] != num and
num_map[y][x - 1] != 0):
24                 return False
25         elif dir == 'U':
26             if num_map[y][x] == num and y == 0:
27                 return False
28             if num_map[y][x] == num and (num_map[y - 1][x] != num and
num_map[y - 1][x] != 0):
29                 return False
30         elif dir == 'D':
31             if num_map[y][x] == num and y == height - 1:
32                 return False
33             if num_map[y][x] == num and (num_map[y + 1][x] != num and
num_map[y + 1][x] != 0):
34                 return False
35         return True
36
37     # move the number to the direction
38     def move(num_map, num, dir):
39         ret_map = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0,
0, 0, 0]]
40         for y in range(height):
41             for x in range(width):
42                 if num_map[y][x] != num and num_map[y][x] != 0:
43                     ret_map[y][x] = num_map[y][x]
44                 elif num_map[y][x] == num:
45                     if dir == 'L':
46                         ret_map[y][x - 1] = num
47                     elif dir == 'R':
48                         ret_map[y][x + 1] = num
49                     elif dir == 'U':
50                         ret_map[y - 1][x] = num
51                     elif dir == 'D':
52                         ret_map[y + 1][x] = num
53         return ret_map
54
55     def DLS(num_map, limit, sol):
56         if limit == 0 and num_map == ans_map:
57             return True, sol
58         elif limit == 0:
59             return False, sol
60
61         possible_move = []
62         # find all possible moves in number 1-10
63         for num in range(1, 11):
64             if checkMove(num_map, num, 'R'):
65                 possible_move.append([str(num), 'R'])
66             if checkMove(num_map, num, 'L'):
67                 possible_move.append([str(num), 'L'])

```

```

68         if checkMove(num_map, num, 'U'):
69             possible_move.append([str(num), 'U'])
70         if checkMove(num_map, num, 'D'):
71             possible_move.append([str(num), 'D'])
72         # check every possible move can lead to final answer or not
73         for act in possible_move:
74             next_num_map = move(num_map, int(act[0]), act[1])
75             fix, ret_sol = DLS(next_num_map, limit - 1, sol + [act])
76             if fix:
77                 return True, ret_sol
78
79         return False, sol
80
81
82     def IDS(num_map):
83         for i in range(0, max_depth):
84             fix, sol = DLS(num_map, i, [])
85             if fix:
86                 return sol
87         return []
88
89     # reading input
90     print('\nPlease enter your input file name(ex. input.txt): ', end='')
91     input_path = input()
92
93     start = time.time()
94     with open(input_path, 'r') as f:
95         tmp = f.read()
96         input_list = tmp.split('\n')
97         for i in range(len(input_list)):
98             if i == 0:
99                 height = int(input_list[0][0:2])
100                 width = int(input_list[0][2:4])
101             else:
102                 l = []
103                 for j in range(0, len(input_list[i]), 2):
104                     num = int(input_list[i][j:j + 2])
105                     l.append(num)
106                 num_map.append(l)
107
108     now_time = time.time()
109
110     print('complete reading input, use time: ' + str(now_time - start) + '
seconds')
111
112     print('\nstarting ids, static ans: ')
113     for i in ans_map:
114         for j in i:
115             print(j, end = ' ')
116         print()
117     print()
118
119     sol = IDS(num_map)
120     print('finish IDS\n')
121     end = time.time()
122
123     with open('output.txt', 'w') as f:
124         f.write('Total run time = ' + str(end - start) + ' seconds.\n')

```

```

125     print('Total run time = ' + str(end - start) + ' seconds.')
126
127     if len(sol) == 0:
128         f.write('There\'s no optimal solution in ' + str(max_depth) + '
moves\n')
129         print('There\'s no optimal solution in ' + str(max_depth) + '
moves')
130     else:
131         f.write('An optimal solution has ' + str(len(sol)) + ' moves:\n')
132         print('An optimal solution has ' + str(len(sol)) + ' moves:')
133         for i in sol:
134             f.write(str(i[0])+i[1]+' ')
135             print(str(i[0])+i[1]+' ', end = '')
136         print()

```

時間複雜度為  $O(b^d)$ ，空間複雜度為  $O(b \times d)$

使用到的資料結構只有 list。

流程為每次找尋盤面上可以移動的方塊，並儲存在一個 list 裡，接著對於每個可以移動的動作進行下一步的探索。由於會造訪所有可能的移動方塊，且深度由淺至深，所以結果必為 optimal，一定能找到最小步數。

上方 source code 裡 max\_depth = 12 是為了方便展示，由於時間複雜度很大，若用太多層時間會不夠跑，以下是針對每種測資的結果。

easy:

```

λ python3 ids.py

Please enter your input file name(ex. input.txt): easy_input.txt
complete reading input, use time: 0.0 seconds

starting ids, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDS

Total run time = 0.005044221878051758 seconds.
An optimal solution has 4 moves:
5R 4U 7L 8D

```

medium:

```
λ python3 ids.py
```

```
Please enter your input file name(ex. input.txt): medium_input.txt  
complete reading input, use time: 0.0 seconds
```

```
starting ids, static ans:
```

```
1 2 2 3
```

```
1 2 2 3
```

```
4 5 5 6
```

```
4 0 0 6
```

```
7 8 9 10
```

```
finish IDS
```

```
Total run time = 0.05189371109008789 seconds.
```

```
An optimal solution has 9 moves:
```

```
1L 2L 3U 3U 6R 5R 4U 7L 8D
```

hard:

```
λ python3 ids.py
```

```
Please enter your input file name(ex. input.txt): hard_input.txt  
complete reading input, use time: 0.0 seconds
```

```
starting ids, static ans:
```

```
1 2 2 3
```

```
1 2 2 3
```

```
4 5 5 6
```

```
4 0 0 6
```

```
7 8 9 10
```

```
finish IDS
```

```
Total run time = 109.48106980323792 seconds.
```

```
An optimal solution has 11 moves:
```

```
3U 6U 8D 5R 4U 7L 9D 8L 10R 9R 8D
```

no answer:

```

λ python3 ids.py

Please enter your input file name(ex. input.txt): no_answer_input.txt
complete reading input, use time: 0.0009992122650146484 seconds

starting ids, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDS

Total run time = 192.04747772216797 seconds.
There's no optimal solution in 12 moves

```

## 4. IDA\*

使用方式：

1. terminal 輸入 python3 ida.py 後會跳出以下訊息

```
Please enter your input file name(ex. input.txt):
```

2. 按照要求輸入輸入檔名稱

```
Please enter your input file name(ex. input.txt): easy_input.txt
```

3. 輸出結果，結果會自動存取於 output.txt

```

λ python3 ida.py

Please enter your input file name(ex. input.txt): easy_input.txt
complete reading input, use time: 0.0 seconds

starting ida, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDA

Total run time = 0.0010004043579101562 seconds.
An optimal solution has 4 moves:
5R 4U 7L 8D

```

source code:

```

1 import time
2
3 num_map = []
4 ans_map = [[1, 2, 2, 3], [1, 2, 2, 3], [4, 5, 5, 6], [4, 0, 0, 6], [7, 8,
9, 10]]

```

```

5  ans_coor = [[0, 0], [0, 1], [0, 3], [2, 0], [2, 1], [2, 3], [4, 0], [4, 1],
6  [4, 2], [4, 3]]
7  # decide max depth for IDA
8  max_limit = 12
9
10 height = 0
11 width = 0
12
13 # count the H() of the num_map
14 def getH(num_map):
15     num_apper = [False]*10
16     cnt = 0
17     for y in range(height):
18         for x in range(width):
19             num = num_map[y][x]
20             if num != 0:
21                 if num_apper[num - 1] == False:
22                     num_apper[num - 1] = True
23                     coor = ans_coor[num - 1]
24                     cnt += abs(y - coor[0]) + abs(x - coor[1])
25
26     return cnt
27
28 # check if the move is valid
29 def checkMove(num_map, num, dir):
30     for y in range(height):
31         for x in range(width):
32             if dir == 'R':
33                 if num_map[y][x] == num and x == width - 1:
34                     return False
35                 if num_map[y][x] == num and (num_map[y][x + 1] != num and
36 num_map[y][x + 1] != 0):
37                     return False
38             elif dir == 'L':
39                 if num_map[y][x] == num and x == 0:
40                     return False
41                 if num_map[y][x] == num and (num_map[y][x - 1] != num and
42 num_map[y][x - 1] != 0):
43                     return False
44             elif dir == 'U':
45                 if num_map[y][x] == num and y == 0:
46                     return False
47                 if num_map[y][x] == num and (num_map[y - 1][x] != num and
48 num_map[y - 1][x] != 0):
49                     return False
50             elif dir == 'D':
51                 if num_map[y][x] == num and y == height - 1:
52                     return False
53                 if num_map[y][x] == num and (num_map[y + 1][x] != num and
54 num_map[y + 1][x] != 0):
55                     return False
56
57     return True
58
59 # move the number to the direction
60 def move(num_map, num, dir):
61     ret_map = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0,
62 0, 0, 0]]
63     for y in range(height):

```



```

57         for x in range(width):
58             if num_map[y][x] != num and num_map[y][x] != 0:
59                 ret_map[y][x] = num_map[y][x]
60             elif num_map[y][x] == num:
61                 if dir == 'L':
62                     ret_map[y][x - 1] = num
63                 elif dir == 'R':
64                     ret_map[y][x + 1] = num
65                 elif dir == 'U':
66                     ret_map[y - 1][x] = num
67                 elif dir == 'D':
68                     ret_map[y + 1][x] = num
69         return ret_map
70
71 def DFS(num_map, flimit, sol, g, next_f):
72     h = getH(num_map)
73     F = g + h
74     if num_map == ans_map:
75         return sol, min(next_f, F)
76     if F > flimit:
77         return [], min(next_f, F)
78
79     possible_move = []
80     # find all possible moves in number 1-10
81     for num in range(1, 11):
82         if checkMove(num_map, num, 'R'):
83             possible_move.append([str(num), 'R'])
84         if checkMove(num_map, num, 'L'):
85             possible_move.append([str(num), 'L'])
86         if checkMove(num_map, num, 'U'):
87             possible_move.append([str(num), 'U'])
88         if checkMove(num_map, num, 'D'):
89             possible_move.append([str(num), 'D'])
90     # check every possible move can lead to final answer or not
91     for act in possible_move:
92         next_num_map = move(num_map, int(act[0]), act[1])
93         ret_sol, ret_F = DFS(next_num_map, flimit, sol + [act], g + 1,
94                             next_f)
95         if ret_F <= flimit:
96             return ret_sol, min(ret_F, next_f)
97
98     return sol, min(F, next_f)
99
100
101 def IDA(num_map):
102     flimit = getH(num_map)
103     while True:
104         now_time = time.time()
105         if now_time - start > 200.0:
106             return []
107         sol, new_flimit = DFS(num_map, flimit, [], 0, max_limit)
108         if len(sol) != 0 and new_flimit == flimit:
109             return sol
110         flimit = min(new_flimit, max_limit)
111         if flimit >= max_limit:
112             return []
113

```

```

114 # reading input
115 print('\nPlease enter your input file name(ex. input.txt): ', end='')
116 input_path = input()
117
118 start = time.time()
119 with open(input_path, 'r') as f:
120     tmp = f.read()
121     input_list = tmp.split('\n')
122     for i in range(len(input_list)):
123         if i == 0:
124             height = int(input_list[0][0:2])
125             width = int(input_list[0][2:4])
126         else:
127             l = []
128             for j in range(0, len(input_list[i]), 2):
129                 num = int(input_list[i][j:j + 2])
130                 l.append(num)
131             num_map.append(l)
132
133 now_time = time.time()
134
135 print('complete reading input, use time: ' + str(now_time - start) + '
seconds')
136
137 print('\nstarting ida, static ans: ')
138 for i in ans_map:
139     for j in i:
140         print(j, end = ' ')
141     print()
142 print()
143
144 sol = IDA(num_map)
145 print('finish IDA\n')
146 end = time.time()
147
148 with open('output.txt', 'w') as f:
149     f.write('Total run time = ' + str(end - start) + ' seconds.\n')
150     print('Total run time = ' + str(end - start) + ' seconds.')
151
152     if len(sol) == 0:
153         f.write('Can\'t find optimal solution in 200 seconds\n')
154         print('Can\'t find optimal solution in 200 seconds\n')
155     else:
156         f.write('An optimal solution has ' + str(len(sol)) + ' moves:\n')
157         print('An optimal solution has ' + str(len(sol)) + ' moves:')
158         for i in sol:
159             f.write(str(i[0])+i[1]+' ')
160             print(str(i[0])+i[1]+' ', end = '')
161         print()

```

時間複雜度為  $O(b^d)$ ，空間複雜度為  $O(b^d)$

使用到的資料結構只有 list。

H() 的設定為每個方塊與最終位置的絕對位置距離相加。

流程為每次找尋盤面上可以移動的方塊，並儲存在一個 list 裡，接著對於每個可以移動的動作進行下一步的探索。計算 F() 值若沒有超過 limit，則繼續下一步的探索。

除了設定最大步數外，在遇到無解答案時，用探索的時間做為 cutoff，若超過 200 秒則強制停止。以下是各測資的結果。

easy:

```
λ python3 ida.py

Please enter your input file name(ex. input.txt): easy_input.txt
complete reading input, use time: 0.0 seconds

starting ida, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDA

Total run time = 0.0010004043579101562 seconds.
An optimal solution has 4 moves:
5R 4U 7L 8D
```

medium:

```
λ python3 ida.py

Please enter your input file name(ex. input.txt): medium_input.txt
complete reading input, use time: 0.0 seconds

starting ida, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDA

Total run time = 0.0019948482513427734 seconds.
An optimal solution has 9 moves:
1L 2L 3U 3U 6R 5R 4U 7L 8D
```

hard:

```
C:\Users\W5036\OneDrive\桌面\code\12\1112
λ python3 ida.py

Please enter your input file name(ex. input.txt): hard_input.txt
complete reading input, use time: 0.0 seconds

starting ida, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDA

Total run time = 0.0019941329956054688 seconds.
An optimal solution has 11 moves:
3U 6U 8D 5R 4U 7L 9D 8L 10R 9R 8D
```

no answer:

```
C:\Users\W5036\OneDrive\桌面\code\12\1112
λ python3 ida.py

Please enter your input file name(ex. input.txt): no_answer_input.txt
complete reading input, use time: 0.0 seconds

starting ida, static ans:
1 2 2 3
1 2 2 3
4 5 5 6
4 0 0 6
7 8 9 10

finish IDA

Total run time = 200.00044631958008 seconds.
Can't find optimal solution in 200 seconds
```

## 5.

由於不會根據輸入的長、寬還有數字找到最終盤面，所以統一以作業二範例的方塊和最終盤面當作測資和目標。

沒有進行過多的輸入檢查，所以如果長寬不是  $5 \times 4$ 、或是數字不是 1-10 就會出錯。