

**LAPORAN HASIL TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2021/2022**

**PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN ALGORITMA
BRANCH AND BOUND**



Disusun oleh :

Steven

13520131

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG**

2022

DAFTAR ISI

BAB I DESKRIPSI MASALAH	2
BAB II PENJELASAN ALGORITMA DALAM MENYELESAIKAN 15-PUZZLE	3
BAB III KODE PROGRAM DALAM PYTHON	4
BAB IV BERKAS TEKS BERISI CONTOH 5 BUAH PERSOALAN 15-PUZZLE	25
BAB V SCREENSHOT INPUT-OUTPUT PROGRAM	26
BAB VI TABEL CHECKLIST.....	30
BAB VII LINK GITHUB.....	31

BAB I

DESKRIPSI MASALAH

Buatlah program dalam Java/Python untuk menyelesaikan persoalan 15-Puzzle dengan menggunakan Algoritma *Branch and Bound* seperti pada materi kuliah. Nilai *bound* tiap simpul adalah penjumlahan cost yang diperlukan untuk mencapai suatu simpul x dari akar, dengan taksiran cost simpul x untuk sampai ke goal. Taksiran cost yang digunakan adalah jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir (*goal state*). Untuk semua instansiasi persoalan 15-puzzle, susunan akhir yang diinginkan adalah sebagai berikut:

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

Posisi awal 15-puzzle dibangkitkan secara acak oleh program dan/atau dimasukkan dari file teks.

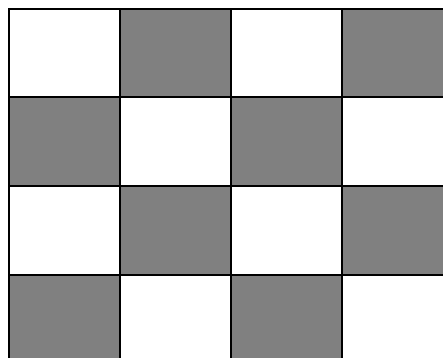
Program harus dapat menentukan apakah posisi awal suatu masukan dapat diselesaikan hingga mencapai susunan akhir, dengan mengimplementasikan fungsi Kurang(i) dan posisi ubin kosong di kondisi awal (X), seperti pada materi kuliah. Jika posisi awal tidak bisa mencapai susunan akhir, program akan menampilkan pesan tidak bisa diselesaikan. Jika dapat diselesaikan, program dapat menampilkan urutan matriks rute (path) aksi yang dilakukan dari posisi awal ke susunan terakhir.

BAB II

PENJELASAN ALGORITMA BRANCH AND BOUND DALAM MENYELESAIKAN 15-PUZZLE

Pada algoritma Branch and Bound, pertama-tama, akan dimasukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi, maka solusi telah ditemukan. Dan, apabila hanya diinginkan satu solusi saja, maka berhentikan program. Langkah keduanya adalah melakukan pengecekan terhadap Q. Apabila Q kosong, maka berhentikan program. Apabila Q tidak kosong, pilih dari antrian Q yang mempunyai nilai cost paling kecil. Apabila terdapat beberapa yang memenuhi, dipilih satu secara sembarang. Langkah berikutnya adalah mengecek apakah simpul yang barusan dipilih merupakan simpul solusi. Apabila simpul merupakan simpul solusi, maka hentikan program apabila hanya satu solusi saja yang diinginkan. Apabila simpul bukan merupakan simpul solusi, maka bangkitkan semua anak-anaknya. Jika simpul ini tidak mempunyai anak, maka kembali ke langkah kedua. Untuk setiap anak, hitung costnya dan masukkan cost tersebut ke dalam Q. Kembali ke langkah kedua dan ulangi lagi langkah-langkahnya.

Untuk persoalan 15-Puzzle ini, simpul solusi hanya dapat tercapai apabila status awal $\sum_{i=0}^{16} KURANG(i) + X$ bernilai genap. Yang dimana $X=1$ jika sel kosong pada posisi awal ada pada sel yang diarsir



Cost setiap simpul dihitung dengan cara menjumlahkan ongkos mencapai simpul dari akar dengan ongkos mencapai simpul tujuan dari simpul

BAB III

KODE PROGRAM DALAM PYTHON

Berikut adalah file program yang diimplementasikan dalam *Python* untuk menyelesaikan persoalan 15-Puzzle dengan memanfaatkan algoritma *Branch and Bound*.

a. board.py

```
board.py X
src > board.py
1 class Board:
2
3     def __init__(self):
4         self.square = []
5         self.missing_number_list = [i for i in range(1, 17)]
6         self.missing_number = 0
7
8     ...
9     override equality function untuk class
10    ...
11    def __eq__(self, other):
12        return self.square == other.square
13
14    ...
15    override hash function untuk class
16    ...
17
18    def __hash__(self):
19        return hash(tuple(self.square))
20
21    ...
22    Fungsi ini berfungsi untuk
23    memasukkan angka ke dalam
24    papan dengan berurut
25    ...
26
27    def add_square(self, new_square):
28        if new_square in self.square:
29            return False
30        self.square.append(new_square)
31        if new_square != -1:
32            self.missing_number_list.remove(new_square)
33        if len(self.missing_number_list) == 1:
34            self.missing_number = self.missing_number_list[0]
35        return True
```

```
board.py X
src > board.py
37     '''
38         fungsi ini berfungsi untuk
39         mengambil angka yang berada
40         pada indeks ke-i di dalam papan.
41         Mengembalikan -1 apabila
42         out of range
43     '''
44     def get_number(self, index):
45         if (index < 1) and (index > 16):
46             return -1
47         return (self.square[index-1])
48
49     '''
50         fungsi ini mengembalikan sebuah indeks yang equivalent dengan format (x, y)
51     '''
52
53     def get_index_xy(self, x, y):
54         return (y * 4) + x
55
56     '''
57         fungsi ini berfungsi untuk
58         mengambil indeks dari angka
59         yang terdapat di dalam papan.
60         Fungsi ini akan mengembalikan nilai
61         -1 apabila tidak terdapat angka tersebut
62         di dalam papan
63     '''
64
65     def get_index(self, number):
66         result = None
67         try:
68             result = self.square.index(number) + 1
69         except ValueError:
70             result = -1
71         return result
```

```

board.py X
src > board.py
73
74     """
75     fungsi ini berfungsi untuk mengembalikan banyaknya
76     angka yang tidak berada pada indeks yang seharusnya
77     """
78     def g_cost(self):
79         result = 0
80         for index in range(len(self.square)):
81             index_check = index + 1
82             elem = self.square[index]
83             if elem == -1:
84                 continue
85             if index_check != elem:
86                 result += 1
87         return result
88
89     """
90     fungsi ini berfungsi untuk mengembalikan copy dari self
91     """
92
93     def copy(self):
94         new_self = Board()
95         for elem in self.square:
96             new_self.add_square(elem)
97         return new_self

```

```

board.py X
src > board.py
98
99     """
100    fungsi ini berfungsi convert
101    class Board ke string
102    """
103
104    def __str__(self):
105        result = ''
106        temp = 0
107        for elem in self.square:
108            if temp == 0:
109                result += '['
110            if elem == -1:
111                result += '__'
112            else:
113                if elem // 10 < 1:
114                    result += '0'
115                result += str(elem)
116            temp += 1
117            if temp == 4:
118                result += ']\n'
119                temp = 0
120            else:
121                result += ' '
122        return result
123
124    def __repr__(self):
125        return self.__str__()
126

```

b. handle_input.py

```
handle_input.py X
src > handle_input.py
1  from board import Board
2  from random import shuffle, randint
3
4  def file_input():
5      print('Please insert the file name')
6      filename = input()
7      board = Board()
8      try:
9          with open(filename, 'r') as f:
10             for line in f:
11                 col_in = line.strip().split(' ')
12                 if len(col_in) != 4:
13                     print('the number of column doesn\'t match the required amount')
14                     return None
15
16                 int_form = []
17
18                 for elem in col_in:
19                     try:
20                         if elem == '_':
21                             int_form.append(-1)
22                         else:
23                             int_form.append(int(elem))
24                             checker = int_form[-1]
25                             if (checker < 1) and (checker > 16):
26                                 print('your input contains a number that is out of range')
27                                 return None
28                     except ValueError:
29                         print('one of your input doesn\'t contain integer')
30                         return None
31
32                 for elem in int_form:
33                     if(not board.add_square(elem)):
34                         print('There\'s a duplicate number in your board')
35                         return None
36             except FileNotFoundError:
37                 print('file not found')
```


🐍 handle_input.py X

src > 🐍 handle_input.py

```
41 def manual_input():
42     print('Please enter the starting matrix.')
43     print('For the empty space, please insert the _ symbol')
44     row_index = 1
45     board = Board()
46     while row_index <= 4:
47         print('please insert the', row_index, 'row')
48
49         ...
50         variable ini digunakan untuk mengecek
51         kondisi dimana apabila input sukses, maka sambung
52         ke baris selanjutnya, dan apabila gagal,
53         mengulangi input baris yang gagal.
54         ...
55
56         succeed = True
57
58         ...
59         kode ini menerima inputan pengguna
60         lalu menghilangkan spasi yang di mana
61         setelah spasi tersebut dihilangkan,
62         dilakukan pemisahan string inputan menjadi
63         beberapa substring menggunakan spasi sebagai
64         pemisah dan mengkonversinya menjadi list
65         sehingga dapat diolah lebih lanjut
66         ...
67
68         user_in = list(input().strip().split(' '))
69
70         ...
71         Code ini mengecek apakah user menginput
72         4 angka
73         ...
74         if len(user_in) != 4:
75             print('Your input didn\'t have enough column')
76             succeed = False
77
78         if not succeed:
79             continue
```

```

handle_input.py X
src > handle_input.py
81     ...
82     Variable ini digunakan untuk
83     menyimpan bentuk integer dari input user
84     ...
85     int_form = []
86
87     ...
88     Code ini berguna untuk
89     mengubah input user
90     menjadi bentuk integer,
91     dan apabila input tersebut merupakan kosong (_)
92     maka diubah menjadi -1 di int_form.
93     Juga mengecek apakah angka yang dimasukkan
94     berada di antara 1-16
95     ...
96     for elem in user_in:
97         try:
98             if elem == '_':
99                 int_form.append(-1)
100            else:
101                int_form.append(int(elem))
102                checker = int_form[-1]
103                if (checker < 1) and (checker > 16):
104                    print('your input contains a number that is out of range')
105                    succeed = False
106            except ValueError:
107                print('one of your input doesn\'t contain integer')
108                succeed = False
109
110        if not succeed:
111            continue

```

```
handle_input.py X
src > handle_input.py
113     ...
114     code ini berfungsi untuk memasukkan
115     baris-baris ke dalam class Board
116     sekaligus mengecek apakah angka tersebut telah ada
117     di dalam class tersebut
118     ...
119     for elem in int_form:
120         if(not board.add_square(elem)):
121             print('There\'s already a', elem, 'in the board')
122             succeed = False
123
124     if not succeed:
125         continue
126
127     if succeed:
128         row_index += 1
129     return board
130
131 def random_input():
132     board_square = [i for i in range(1, 16)]
133     board_square.append(-1)
134     shuffle(board_square)
135     board = Board()
136     for elem in board_square:
137         board.add_square(elem)
138     return board
```

c. cli.py

```
cli.py x
src > cli.py
1  from board import Board
2  from handle_input import manual_input, random_input, file_input
3  from time import time
4  from functools import cmp_to_key
5
6  ...
7      fungsi untuk dimasukkan ke dalam parameter key
8      di dalam fungsi min(), mengembalikan tuple dimana
9      elem[0] merupakan level dan elem[1] merupakan cost
10  ...
11  def bnb_cmp(a, b):
12      if a[0] < b[0]:
13          return 1
14      elif a[0] > b[0]:
15          return -1
16      else:
17          if a[1] > b[1]:
18              return 1
19          elif a[1] < b[1]:
20              return -1
21          else:
22              return 0
23
24  ...
25      fungsi untuk mendapatkan semua angka yang lebih kecil dari x
26      dengan constraint lebih besar dari 1
27  ...
28  def get_smaller_number_list(number):
29      result = []
30      temp = number - 1
31      while temp != 0:
32          result.append(temp)
33          temp -= 1
34      return result
```

```

cli.py ×
src > cli.py
36 '''
37     fungsi ini untuk mendapatkan apakah kurang nanti akan
38     ditambah 1 atau 0. Fungsi ini mendapatkan hasil tersebut dengan pattern
39     apabila index dari kosong yang terdapat pada board tersebut (asumsi mulai dari 0),
40     dibagi 4 dan dimod 2 hasilnya sama dengan dimod 2 saja, maka tidak berada di area arsir, sebaliknya
41     apabila hasilnya dibagi 4 dan dimod 2 tidak sama dengan dimod 2 saja, maka berada di area arsir.
42 '''
43 def get_kurang_x(board):
44     position = board.get_index(-1)
45     position -= 1
46     row_check = (position // 4) % 2
47     column_check = position % 2
48     if row_check == column_check:
49         return 0
50     else:
51         return 1
52
53 '''
54     fungsi ini mengkalkulasi semua kurang()
55     dan mengembalikannya dalam bentuk list
56 '''
57 def get_kurang(board):
58     result = []
59     for index in range(len(board.square)):
60         elem = board.square[index]
61         if elem == -1:
62             elem = board.missing_number
63             smaller_list = get_smaller_number_list(elem)
64             count = 0
65             for i in smaller_list:
66                 other_index = board.get_index(i)
67                 if other_index > index:
68                     count += 1
69             result.append((elem, count))
70     result.sort(key=lambda x: x[0])
71     return result

```

```
cli.py ×
src > cli.py
73     '''
74     fungsi ini berfungsi untuk memprint out
75     kurang dalam format "i: {elem} kurang(i): {kurang}"
76     '''
77     def print_kurang(kurang):
78         for elem in kurang:
79             print('i:', elem[0], 'kurang(i):', elem[1])
80
81     '''
82     pruning setelah mendapatkan solusi
83     dari algoritma branch and bound
84     '''
85     def prune(queue):
86         queue.empty()
87
88     '''
89     fungsi untuk membuat branch and bound
90     '''
91     def bnb(board):
92         print(board)
93
94         kurang = get_kurang(board)
95         print_kurang(kurang)
96
97         kurang_sum = 0
98         for elem in kurang:
99             kurang_sum += elem[1]
100
101         x = get_kurang_x(board)
102
103         checker = kurang_sum + x
104         print('Sum from i=0 to 16 of KURANG(i) + x = ', checker)
105         if checker % 2 != 0:
106             print('Status tujuan tidak dapat dicapai (Puzzle Unsolvble)')
107             return (None, None)
108
109         queue = [(0, 0, board, None)]
110         set_board = {board}
```

```
cli.py ×
src > cli.py
112     print('\nDoing Branch and Bound')
113     print('-----')
114
115     prev_level = 0
116
117     total_explored = 1
118
119     board_output = []
120     while True:
121
122         to_check = min(queue, key=cmp_to_key(bnb_cmp))
123         queue.remove(to_check)
124         check_board = to_check[2]
125         check_level = to_check[0] + 1
126         check_dir = to_check[3]
127
128         if prev_level >= check_level:
129             board_output = [(board, None)]
130
131         prev_level = check_level
132
133         board_output.append((check_board, check_dir))
134
135         if check_board.g_cost() == 0:
136             break
137
138         ...
139         |     mendapatkan posisi dari square yang kosong di board
140         |     ...
141
142         empty_pos = check_board.square.index(-1)
143         row_pos = empty_pos // 4
144         col_pos = empty_pos % 4
```

```
cli.py ×
src > cli.py
146     ...
147     move up
148     ...
149
150     succeed = True
151     up_board = check_board.copy()
152     if row_pos == 0:
153         succeed = False
154     else:
155         up_row = row_pos - 1
156         up_index = up_board.get_index_xy(col_pos, up_row)
157
158         temp = up_board.square[up_index]
159         up_board.square[up_index] = up_board.square[empty_pos]
160         up_board.square[empty_pos] = temp
161
162     if up_board in set_board:
163         succeed = False
164
165     if succeed:
166         cost = up_board.g_cost() + (check_level)
167         queue.append((check_level, cost, up_board, 'UP'))
168         set_board.add(up_board)
169         total_explored += 1
```

```
cli.py ×
src > cli.py
170
171     ...
172     move down
173     ...
174
175     succeed = True
176     bottom_board = check_board.copy()
177
178     if row_pos == 3:
179         succeed = False
180     else:
181         bottom_row = row_pos + 1
182         bottom_index = bottom_board.get_index_xy(col_pos, bottom_row)
183
184         temp = bottom_board.square[bottom_index]
185         bottom_board.square[bottom_index] = bottom_board.square[empty_pos]
186         bottom_board.square[empty_pos] = temp
187
188     if bottom_board in set_board:
189         succeed = False
190
191     if succeed:
192         cost = bottom_board.g_cost() + (check_level)
193         queue.append((check_level, cost, bottom_board, 'DOWN'))
194         set_board.add(bottom_board)
195         total_explored += 1
```



```
cli.py x
src > cli.py
197     ...
198     move left
199     ...
200
201     succeed = True
202     left_board = check_board.copy()
203
204     if col_pos == 0:
205         succeed = False
206     else:
207         left_pos = col_pos - 1
208         left_index = left_board.get_index_xy(left_pos, row_pos)
209
210         temp = left_board.square[left_index]
211         left_board.square[left_index] = left_board.square[empty_pos]
212         left_board.square[empty_pos] = temp
213
214     if left_board in set_board:
215         succeed = False
216
217     if succeed:
218         cost = left_board.g_cost() + (check_level)
219         queue.append((check_level, cost, left_board, 'LEFT'))
220         set_board.add(left_board)
221         total_explored += 1
```

```
cli.py x
src > cli.py
223     ...
224     move right
225     ...
226
227     succeed = True
228     right_board = check_board.copy()
229
230     if col_pos == 3:
231         succeed = False
232     else:
233         right_pos = col_pos + 1
234         right_index = right_board.get_index_xy(right_pos, row_pos)
235
236         temp = right_board.square[right_index]
237         right_board.square[right_index] = right_board.square[empty_pos]
238         right_board.square[empty_pos] = temp
239
240     if right_board in set_board:
241         succeed = False
242
243     if succeed:
244         cost = right_board.g_cost() + (check_level)
245         queue.append((check_level, cost, right_board, 'RIGHT'))
246         set_board.add(right_board)
247         total_explored += 1
248
249     return (board_output, total_explored)
```

```
cli.py ×
src > cli.py
254 def main():
255
256     print('Please choose an input type')
257     print('1. Manual Input')
258     print('2. Random Input')
259     print('3. File Input')
260     input_type = int(input('>> '))
261
262     board = None
263     if input_type == 1:
264         board = manual_input()
265     elif input_type == 2:
266         board = random_input()
267     elif input_type == 3:
268         board = file_input()
269     else:
270         main()
271     return
272
273     if board is None:
274         return
275
276     start = time()
277
278     (board_output, total_explored) = bnb(board)
279
280     end = time()
281
282     if board_output is not None:
283         for elem in board_output:
284             if elem[1] != None:
285                 print(elem[1])
286             print(elem[0])
287
288     if total_explored is not None:
289         print('Jumlah simpul yang dibangkitkan:', total_explored)
290         print('Total Move Taken:', len(board_output)-1)
291     print('This program takes ', end - start, ' second(s)')
292
293 if __name__ == '__main__':
294     main()
```

d. app.py

```
app.py ×
src > app.py
1  import eel
2  from board import Board
3  from time import time
4  from functools import cmp_to_key
5
6  '''
7      fungsi untuk dimasukkan ke dalam parameter key
8      di dalam fungsi min(), mengembalikan tuple dimana
9      elem[0] merupakan level dan elem[1] merupakan cost
10 '''
11 def bnb_cmp(a, b):
12     if a[0] < b[0]:
13         return 1
14     elif a[0] > b[0]:
15         return -1
16     else:
17         if a[1] > b[1]:
18             return 1
19         elif a[1] < b[1]:
20             return -1
21         else:
22             return 0
23
24 '''
25     fungsi untuk mendapatkan semua angka yang lebih kecil dari x
26     dengan constraint lebih besar dari 1
27 '''
28 def get_smaller_number_list(number):
29     result = []
30     temp = number - 1
31     while temp != 0:
32         result.append(temp)
33         temp -= 1
34     return result
```

```

app.py ×
src > app.py
36
37     '''
38     fungsi ini untuk mendapatkan apakah kurang nanti akan
39     ditambah 1 atau 0. Fungsi ini mendapatkan hasil tersebut dengan patterm
40     apabila index dari kosong yang terdapat pada board tersebut (asumsi mulai dari 0),
41     dibagi 4 dan dimod 2 hasilnya sama dengan dimod 2 saja, maka tidak berada di area arsir, sebaliknya
42     apabila hasilnya dibagi 4 dan dimod 2 tidak sama dengan dimod 2 saja, maka berada di area arsir.
43     '''
44     def get_kurang_x(board):
45         position = board.get_index(-1)
46         position -= 1
47         row_check = (position // 4) % 2
48         column_check = position % 2
49         if row_check == column_check:
50             return 0
51         else:
52             return 1
53     '''
54     fungsi ini mengkalkulasi semua kurang()
55     dan mengembalikannya dalam bentuk list
56     '''
57     def get_kurang(board):
58         result = []
59         for index in range(len(board.square)):
60             elem = board.square[index]
61             if elem == -1:
62                 elem = board.missing_number
63                 smaller_list = get_smaller_number_list(elem)
64                 count = 0
65                 for i in smaller_list:
66                     other_index = board.get_index(i)
67                     if other_index > index:
68                         count += 1
69                 result.append((elem, count))
70         result.sort(key=lambda x: x[0])
71         return result

```

```

app.py ×
src > app.py
72
73     '''
74     pruning setelah mendapatkan solusi
75     dari algoritma branch and bound
76     '''
77     def prune(queue):
78         queue.empty()

```

```
app.py ×
src > app.py
79
80     ...
81     fungsi untuk membuat branch and bound
82     ...
83     def bnb(board):
84         # result = {}
85
86         kurang = get_kurang(board)
87         # result['kurang'] = kurang
88         eel.set_kurang(kurang)
89
90         kurang_sum = 0
91         for elem in kurang:
92             kurang_sum += elem[1]
93
94         x = get_kurang_x(board)
95
96         checker = kurang_sum + x
97         eel.set_kurang_x(checker)
98         if checker % 2 != 0:
99             # result['error'] = 'Unable to finish the 15 puzzle'
100             eel.set_error('Unable to finish the 15 puzzle')
101             return (None, None)
102
103         queue = [(0, 0, board)]
104         set_board = {board}
105
106         # result['boards'] = []
107
108         board_id = 1
109
110         output_board = []
111         prev_level = 0
112
113         total_explored = 1
```

```
app.py X
src > app.py
115     while True:
116
117         to_check = min(queue, key=cmp_to_key(bnb_cmp))
118         queue.remove(to_check)
119         check_board = to_check[2]
120         check_level = to_check[0] + 1
121
122         # result['boards'].append({'board_id': board_id, 'board':check_board.square})
123         if prev_level >= check_level:
124             output_board = [board]
125
126         output_board.append(check_board)
127
128         prev_level = check_level
129
130         if check_board.g_cost() == 0:
131             break
132
133         ...
134         |   mendapatkan posisi dari square yang kosong di board
135         ...
136
137         empty_pos = check_board.square.index(-1)
138         row_pos = empty_pos // 4
139         col_pos = empty_pos % 4
```

```
app.py X
src > app.py
141     ...
142     |   move up
143     ...
144
145     succeed = True
146     up_board = check_board.copy()
147     if row_pos == 0:
148         succeed = False
149     else:
150         up_row = row_pos - 1
151         up_index = up_board.get_index_xy(col_pos, up_row)
152
153         temp = up_board.square[up_index]
154         up_board.square[up_index] = up_board.square[empty_pos]
155         up_board.square[empty_pos] = temp
156
157         if up_board in set_board:
158             succeed = False
159
160         if succeed:
161             cost = up_board.g_cost() + (check_level)
162             queue.append((check_level, cost, up_board))
163             set_board.add(up_board)
164             total_explored += 1
```

```
app.py ×
src > app.py
166     ...
167     |     move bottom
168     |     ...
169
170     succeed = True
171     bottom_board = check_board.copy()
172
173     if row_pos == 3:
174         succeed = False
175     else:
176         bottom_row = row_pos + 1
177         bottom_index = bottom_board.get_index_xy(col_pos, bottom_row)
178
179         temp = bottom_board.square[bottom_index]
180         bottom_board.square[bottom_index] = bottom_board.square[empty_pos]
181         bottom_board.square[empty_pos] = temp
182
183     if bottom_board in set_board:
184         succeed = False
185
186     if succeed:
187         cost = bottom_board.g_cost() + (check_level)
188         queue.append((check_level, cost, bottom_board))
189         set_board.add(bottom_board)
190         total_explored += 1
```

```
app.py ×
src > app.py
192     ...
193     |     move left
194     |     ...
195
196     succeed = True
197     left_board = check_board.copy()
198
199     if col_pos == 0:
200         succeed = False
201     else:
202         left_pos = col_pos - 1
203         left_index = left_board.get_index_xy(left_pos, row_pos)
204
205         temp = left_board.square[left_index]
206         left_board.square[left_index] = left_board.square[empty_pos]
207         left_board.square[empty_pos] = temp
208
209     if left_board in set_board:
210         succeed = False
211
212     if succeed:
213         cost = left_board.g_cost() + (check_level)
214         queue.append((check_level, cost, left_board))
215         set_board.add(left_board)
216         total_explored += 1
```

```
app.py ×
src > app.py
218     ...
219     |     move right
220     |     ...
221
222     succeed = True
223     right_board = check_board.copy()
224
225     if col_pos == 3:
226     |     succeed = False
227     else:
228     |     right_pos = col_pos + 1
229     |     right_index = right_board.get_index_xy(right_pos, row_pos)
230
231     |     temp = right_board.square[right_index]
232     |     right_board.square[right_index] = right_board.square[empty_pos]
233     |     right_board.square[empty_pos] = temp
234
235     if right_board in set_board:
236     |     succeed = False
237
238     if succeed:
239     |     cost = right_board.g_cost() + (check_level)
240     |     queue.append((check_level, cost, right_board))
241     |     set_board.add(right_board)
242     |     total_explored += 1
243
244     return (output_board, total_explored)
```



```
app.py ×
src > app.py
246 def main():
247     eel.init('dist')
248
249     @eel.expose
250     def calculate_bnb(board_val):
251         board = Board()
252         for elem in board_val:
253             board.add_square(elem)
254         print('calculating')
255
256         start = time()
257         (output_board, total_explored) = bnb(board)
258         end = time()
259         eel.set_time_taken(end - start)
260         if total_explored is not None:
261             eel.set_explored(total_explored)
262         output_board_list = []
263         if output_board is not None:
264             for elem in output_board:
265                 output_board_list.append(elem.square)
266         return output_board_list
267
268
269     @eel.expose
270     def stop_program():
271         exit()
272
273     #eel.start('index.html', mode='edge')
274     #eel.start('index.html', mode='chrome')
275     eel.start('index.html')
276
277
278 if __name__ == '__main__':
279     main()
```

BAB IV

BERKAS TEKS BERISI CONTOH 5 BUAH PERSOALAN 15-PUZZLE

```
test1.txt - Notepad
File Edit View
1 2 3 4
5 6 _ 8
9 10 7 11
13 14 15 12
Ln 4, Col 12 100% Windows (CRLF) UTF-8
```

```
test2.txt - Notepad
File Edit View
1 3 4 15
2 _ 5 12
7 6 11 14
8 9 10 13
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

```
test3.txt - Notepad
File Edit View
1 3 6 4
5 2 _ 8
9 10 7 11
13 14 15 12
Ln 4, Col 12 100% Windows (CRLF) UTF-8
```

```
test4.txt - Notepad
File Edit View
1 11 9 13
3 2 _ 10
8 12 14 15
1 5 6 4
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

```
test5.txt - Notepad
File Edit View
_ 3 6 4
1 2 8 11
5 9 7 12
13 10 14 15
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

BAB V

SCREENSHOT INPUT-OUTPUT PROGRAM

test1.txt

```
Please choose an input type
1. Manual Input
2. Random Input
3. File Input
>> 3
Please insert the file name
../test/test1.txt
[01 02 03 04]
[05 06 _ 08]
[09 10 07 11]
[13 14 15 12]

i: 1 kurang(i): 0
i: 2 kurang(i): 0
i: 3 kurang(i): 0
i: 4 kurang(i): 0
i: 5 kurang(i): 0
i: 6 kurang(i): 0
i: 7 kurang(i): 0
i: 8 kurang(i): 1
i: 9 kurang(i): 1
i: 10 kurang(i): 1
i: 11 kurang(i): 0
i: 12 kurang(i): 0
i: 13 kurang(i): 1
i: 14 kurang(i): 1
i: 15 kurang(i): 1
i: 16 kurang(i): 9
Sum from i=0 to 16 of KURANG(i) + X = 16
```

```
Doing Branch and Bound
-----
[01 02 03 04]
[05 06 _ 08]
[09 10 07 11]
[13 14 15 12]

DOWN
[01 02 03 04]
[05 06 07 08]
[09 10 _ 11]
[13 14 15 12]

RIGHT
[01 02 03 04]
[05 06 07 08]
[09 10 11 _]
[13 14 15 12]

DOWN
[01 02 03 04]
[05 06 07 08]
[09 10 11 12]
[13 14 15 _]

Jumlah simpul yang dibangkitkan: 10
Total Move Taken: 3
This program takes 0.000997781753540039 second(s)
```

test2.txt

```
Please choose an input type
1. Manual Input
2. Random Input
3. File Input
>> 3
Please insert the file name
../test/test2.txt
[01 03 04 15]
[02 _ 05 12]
[07 06 11 14]
[08 09 10 13]

i: 1 kurang(i): 0
i: 2 kurang(i): 0
i: 3 kurang(i): 1
i: 4 kurang(i): 1
i: 5 kurang(i): 0
i: 6 kurang(i): 0
i: 7 kurang(i): 1
i: 8 kurang(i): 0
i: 9 kurang(i): 0
i: 10 kurang(i): 0
i: 11 kurang(i): 3
i: 12 kurang(i): 6
i: 13 kurang(i): 0
i: 14 kurang(i): 4
i: 15 kurang(i): 11
i: 16 kurang(i): 10
Sum from i=0 to 16 of KURANG(i) + X = 37
Status tujuan tidak dapat dicapai (Puzzle Unsolvble)
This program takes 0.0009982585906982422 second(s)
```

test3.txt

```
Please choose an input type
1. Manual Input
2. Random Input
3. File Input
>> 3
Please insert the file name
../test/test3.txt
[01 03 06 04]
[05 02 __ 08]
[09 10 07 11]
[13 14 15 12]

i: 1 kurang(i): 0
i: 2 kurang(i): 0
i: 3 kurang(i): 1
i: 4 kurang(i): 1
i: 5 kurang(i): 1
i: 6 kurang(i): 3
i: 7 kurang(i): 0
i: 8 kurang(i): 1
i: 9 kurang(i): 1
i: 10 kurang(i): 1
i: 11 kurang(i): 0
i: 12 kurang(i): 0
i: 13 kurang(i): 1
i: 14 kurang(i): 1
i: 15 kurang(i): 1
i: 16 kurang(i): 9
Sum from i=0 to 16 of KURANG(i) + X = 22

Doing Branch and Bound
-----
[01 03 06 04]
[05 02 __ 08]
[09 10 07 11]
[13 14 15 12]

DOWN
[01 03 06 04]
[05 02 07 08]
[09 10 __ 11]
[13 14 15 12]
```

```
RIGHT
[01 03 06 04]
[05 02 07 08]
[09 10 11 __]
[13 14 15 12]

DOWN
[01 03 06 04]
[05 02 07 08]
[09 10 11 12]
[13 14 15 __]

LEFT
[01 03 06 04]
[05 02 07 08]
[09 10 11 12]
[13 14 __ 15]

UP
[01 03 06 04]
[05 02 07 08]
[09 10 __ 12]
[13 14 11 15]

UP
[01 03 06 04]
[05 02 __ 08]
[09 10 07 12]
[13 14 11 15]

UP
[01 03 __ 04]
[05 02 06 08]
[09 10 07 12]
[13 14 11 15]
```

```
LEFT
[01 __ 03 04]
[05 02 06 08]
[09 10 07 12]
[13 14 11 15]

DOWN
[01 02 03 04]
[05 __ 06 08]
[09 10 07 12]
[13 14 11 15]

RIGHT
[01 02 03 04]
[05 06 __ 08]
[09 10 07 12]
[13 14 11 15]

DOWN
[01 02 03 04]
[05 06 07 08]
[09 10 __ 12]
[13 14 11 15]

DOWN
[01 02 03 04]
[05 06 07 08]
[09 10 11 12]
[13 14 __ 15]

RIGHT
[01 02 03 04]
[05 06 07 08]
[09 10 11 12]
[13 14 15 __]

Jumlah simpul yang dibangkitkan: 34
Total Move Taken: 13
This program takes 0.0040013790130615234 second(s)
```

test4.txt

```
Please choose an input type
1. Manual Input
2. Random Input
3. File Input
>> 3
Please insert the file name
../test/test4.txt
[07 11 09 13]
[03 02 __ 10]
[08 12 14 15]
[01 05 06 04]

i: 1 kurang(i): 0
i: 2 kurang(i): 1
i: 3 kurang(i): 2
i: 4 kurang(i): 0
i: 5 kurang(i): 1
i: 6 kurang(i): 1
i: 7 kurang(i): 6
i: 8 kurang(i): 4
i: 9 kurang(i): 7
i: 10 kurang(i): 5
i: 11 kurang(i): 9
i: 12 kurang(i): 4
i: 13 kurang(i): 9
i: 14 kurang(i): 4
i: 15 kurang(i): 4
i: 16 kurang(i): 9
Sum from i=0 to 16 of KURANG(i) + X = 67
Status tujuan tidak dapat dicapai (Puzzle Unsolvable)
This program takes 0.004004716873168945 second(s)
```

test5.txt

```
Please choose an input type
1. Manual Input
2. Random Input
3. File Input
>> 3
Please insert the file name
../test/test5.txt
[__ 03 06 04]
[01 02 08 11]
[05 09 07 12]
[13 10 14 15]

i: 1 kurang(i): 0
i: 2 kurang(i): 0
i: 3 kurang(i): 2
i: 4 kurang(i): 2
i: 5 kurang(i): 0
i: 6 kurang(i): 4
i: 7 kurang(i): 0
i: 8 kurang(i): 2
i: 9 kurang(i): 1
i: 10 kurang(i): 0
i: 11 kurang(i): 4
i: 12 kurang(i): 1
i: 13 kurang(i): 1
i: 14 kurang(i): 0
i: 15 kurang(i): 0
i: 16 kurang(i): 15
Sum from i=0 to 16 of KURANG(i) + X = 32
```

Doing Branch and Bound

[__ 03 06 04]
[01 02 08 11]
[05 09 07 12]
[13 10 14 15]

DOWN

[01 03 06 04]
[__ 02 08 11]
[05 09 07 12]
[13 10 14 15]

DOWN

[01 03 06 04]
[05 02 08 11]
[__ 09 07 12]
[13 10 14 15]

RIGHT

[01 03 06 04]
[05 02 08 11]
[09 __ 07 12]
[13 10 14 15]

DOWN

[01 03 06 04]
[05 02 08 11]
[09 10 07 12]
[13 __ 14 15]

RIGHT

[01 03 06 04]
[05 02 08 11]
[09 10 07 12]
[13 14 __ 15]

RIGHT

[01 03 06 04]
[05 02 08 11]
[09 10 07 12]
[13 14 15 __]

UP

[01 03 06 04]
[05 02 08 11]
[09 10 07 __]
[13 14 15 12]

UP

[01 03 06 04]
[05 02 08 __]
[09 10 07 11]
[13 14 15 12]

LEFT

[01 03 06 04]
[05 02 __ 08]
[09 10 07 11]
[13 14 15 12]

DOWN

[01 03 06 04]
[05 02 07 08]
[09 10 __ 11]
[13 14 15 12]

RIGHT

[01 03 06 04]
[05 02 07 08]
[09 10 11 __]
[13 14 15 12]

```
DOWN
[01 03 06 04]
[05 02 07 08]
[09 10 11 12]
[13 14 15 __]
```

```
LEFT
[01 03 06 04]
[05 02 07 08]
[09 10 11 12]
[13 14 __ 15]
```

```
UP
[01 03 06 04]
[05 02 07 08]
[09 10 __ 12]
[13 14 11 15]
```

```
UP
[01 03 06 04]
[05 02 __ 08]
[09 10 07 12]
[13 14 11 15]
```

```
UP
[01 03 __ 04]
[05 02 06 08]
[09 10 07 12]
[13 14 11 15]
```

```
LEFT
[01 __ 03 04]
[05 02 06 08]
[09 10 07 12]
[13 14 11 15]
```

```
DOWN
[01 02 03 04]
[05 __ 06 08]
[09 10 07 12]
[13 14 11 15]
```

```
RIGHT
[01 02 03 04]
[05 06 __ 08]
[09 10 07 12]
[13 14 11 15]
```

```
DOWN
[01 02 03 04]
[05 06 07 08]
[09 10 __ 12]
[13 14 11 15]
```






```
DOWN
[01 02 03 04]
[05 06 07 08]
[09 10 11 12]
[13 14 __ 15]
```

```
RIGHT
[01 02 03 04]
[05 06 07 08]
[09 10 11 12]
[13 14 15 __]
```

```
Jumlah simpul yang dibangkitkan:
Total Move Taken: 22
This program takes 0.00600028038
```

BAB VI

Tabel Checklist

Poin	Ya	Tidak
Program berhasil dikompilasi		
Program berhasil running		
Program dapat menerima input dan menulis output		
Luaran sudah benar untuk semua data uji		
Bonus dibuat		

BAB VII
LINK GITHUB

https://github.com/loopfree/Tucil3_13520131