

Tugas Kecil Strategi Algoritma – IF2211

**Penyelesaian Persoalan 15-Puzzle dengan
Algoritma *Branch and Bound***



13520048 – Arik Rayi Arkananta

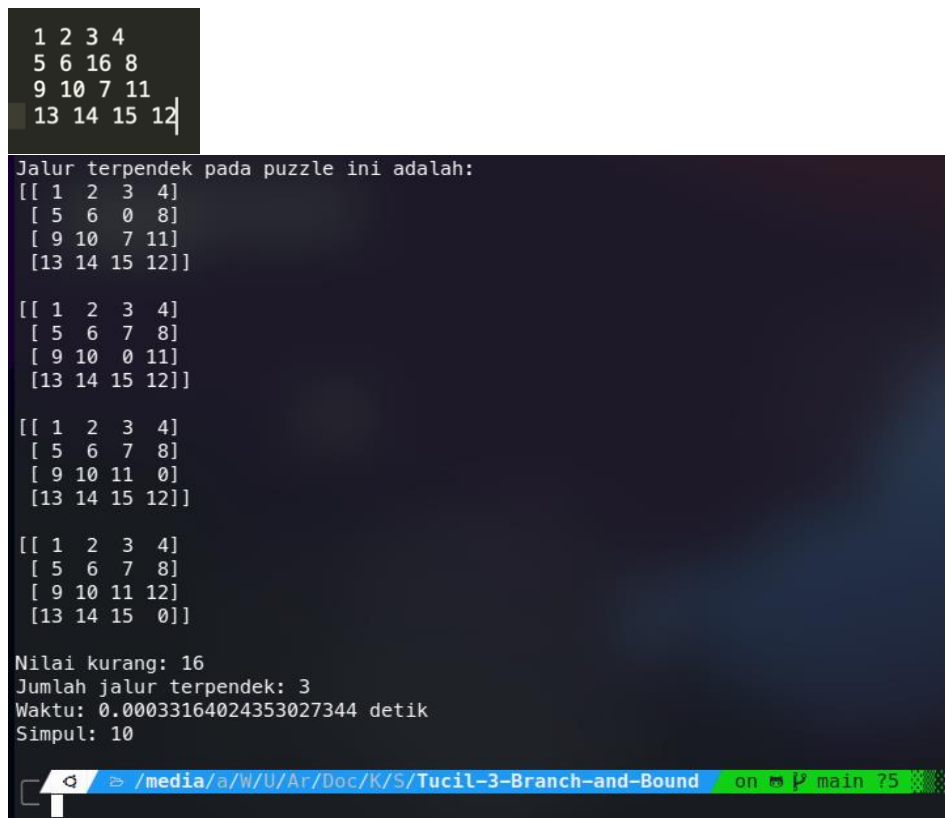
A. Algoritma *Branch and Bound*

Pada program yang telah saya buat, digunakan algoritma *Branch and Bound* dengan bahasa pemrograman python untuk mencari penyelesaian dari persoalan 15-Puzzle. Berikut adalah cara kerja algoritma *Branch and Bound* dari sebuah tuple 15-Puzzle yang berisi ksuatu 15-Puzzle:

1. Pertama dibuat suatu priorityQueue yang akan diisi oleh node-node pergerakan dari 15-Puzzle berdasarkan cost-nya masing-masing dan suatu set Visited yang akan menyimpan semua kondisi puzzle yang sudah pernah dilewati
2. Hitung nilai kurang dari kondisi awal 15-Puzzle, jika hasilnya genap maka lanjut ke nomor 3, jika hasilnya ganjil maka program akan berhenti karena tidak bisa dilakukan pencarian untuk penyelesaiannya
3. Masukkan kondisi awal 15-Puzzle ke dalam priorityQueue beserta dengan cost-nya dan diurutkan berdasarkan cost-nya. Lalu, jika belum ada di set Visited masukan juga kondisi matrix puzzle ke set Visited
4. Cari semua kemungkinan pergerakan puzzle yang mungkin dari hasil dequeue priorityQueue, seperti ke atas, kanan, kiri, dan bawah
5. Untuk tiap kemungkinan, jika memenuhi penyelesaian akhir, program akan mengembalikan hasil puzzle. Jika tidak, maka akan dicek apakah sudah pernah dilewati di set Visited, jika sudah maka tidak akan dienqueue ke priorityQueue, jika belum ada maka akan dienqueue ke priorityQueue
6. Ulangi dari langkah keempat sampai menemui penyelesaian akhir

B. Screenshot Input dan Output

1. Testcase 1



```
1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12

Jalur terpendek pada puzzle ini adalah:
[[ 1  2  3  4]
 [ 5  6  0  8]
 [ 9 10  7 11]
 [13 14 15 12]]

[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10  0 11]
 [13 14 15 12]]

[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11  0]
 [13 14 15 12]]

[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15  0]]

Nilai kurang: 16
Jumlah jalur terpendek: 3
Waktu: 0.00033164024353027344 detik
Simpul: 10
```

The screenshot shows a terminal window with a dark background. At the top, a 4x4 matrix of numbers is displayed. Below it, the text "Jalur terpendek pada puzzle ini adalah:" is followed by three 4x4 matrices, each representing a different shortest path solution. The matrices show the sequence of numbers in the puzzle state. At the bottom, summary statistics are provided: "Nilai kurang: 16", "Jumlah jalur terpendek: 3", "Waktu: 0.00033164024353027344 detik", and "Simpul: 10". The terminal window's title bar at the very bottom indicates the file path and the current program being run.

2. Testcase 2

```
1 2 5 4
3 6 16 7
9 10 8 11
13 14 15 12
```

```
[[ 1 2 3 4]
 [ 5 6 0 7]
 [ 9 10 11 8]
 [13 14 15 12]]
```

```
[[ 1 2 3 4]
 [ 5 6 7 0]
 [ 9 10 11 8]
 [13 14 15 12]]
```

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 11 0]
 [13 14 15 12]]
```

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 11 12]
 [13 14 15 0]]
```

Nilai kurang: 18
Jumlah jalur terpendek: 25
Waktu: 1.8728492259979248 detik
Simpul: 293657

🔍 /media/a/W/U/Ar/Doc/K/S/Tucil-3-Branch-and-Bound on 🖨️ main ?6

3. Testcase 3

```
1 2 3 4
5 7 10 8
11 9 6 16
13 14 15 12
```

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 0 10 12]
 [13 14 11 15]]
```

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 0 12]
 [13 14 11 15]]
```

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 11 12]
 [13 14 0 15]]
```

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 11 12]
 [13 14 15 0]]
```

Nilai kurang: 16
Jumlah jalur terpendek: 21
Waktu: 0.09972620010375977 detik
Simpul: 29645

🔍 /media/a/W/U/Ar/Doc/K/S/Tucil-3-Branch-and-Bound on 🖨️ main ?7

4. Testcase 4

```

3 1 5 6
7 8 10 9
11 4 2 16
13 15 14 12

```

Nilai kurang: 27
Cannot be solved

```

❏ /media/a/W/U/Ar/Doc/K/S/Tucil-3-Branch-and-Bound on main ?8

```

5. Testcase 5

```

1 3 5 7
9 11 13 15
16 2 4 6
8 10 12 14

```

Nilai kurang: 35
Cannot be solved

```

❏ /media/a/W/U/Ar/Doc/K/S/Tucil-3-Branch-and-Bound on main ?9

```

C. Checklist Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil running	√	
3. Program dapat menerima input dan menuliskan output.	√	
4. Luaran sudah benar untuk semua data uji	√	
5. Bonus dibuat		√

D. Kode Program

```
1  import numpy as np
2  import heapq as hq
3  import time
4
5  # Class untuk menyimpan tiap simpul
6  class Node:
7      def __init__(self, level, cost, buffer, emptyTile, parent):
8          self.cost = cost
9          self.level = level
10         self.buffer = buffer
11         self.emptyTile = emptyTile
12         self.parent = parent
13
14     def __lt__(self, other):
15         return self.cost <= other.cost
16
17 # Fungsi pengubahan hasil ke bentuk matrix
18 def changeToZero(buffer):
19     matrixZero = buffer.copy()
20     matrixZero.resize(4,4)
21     for r in range(0,4):
22         for c in range(0,4):
23             if matrixZero[r,c] == 16:
24                 matrixZero[r,c] = 0
25     return matrixZero
26
27 # Fungsi pencarian nilai kurang
28 def fungsiKurang(buffer):
29     kurang = 0;
30     for i in range(1,17):
31         x = 0;
32         while (buffer[x] != i):
33             x += 1;
34         for j in range(x,16):
35             if buffer[j] < i:
36                 kurang += 1
37         for r in range(0,4):
38             for c in range(0,4):
39                 if buffer[(r*4)+c] == 16:
40                     if (r + c) % 2 != 0:
41                         kurang += 1
42     return kurang
43
44 # Fungsi menghitung cost dari pergerakan dan cost parent
45 def fungsiCost(parentCost, r, c, element, r1, c1):
46     if ((r*4) + c + 1 == element):
47         return parentCost
48     elif ((r1*4) + c1 + 1 == element):
49         return parentCost + 2
50     else:
51         return parentCost + 1
52
53 # Fungsi membuat tuple baru sesuai pergerakan
54 def move(parent,r,c,r2,c2):
55     tempMatrix = list(parent.buffer)
56     tempMatrix[(r*4)+c] = tempMatrix[(r2*4)+c2]
57     tempMatrix[(r2*4)+c2] = 16
58     return tuple(tempMatrix)
59
60 # Fungsi utama pencarian solusi
61 def cariKemungkinan(prioQueue, visited):
62     nodes = 1
63     while (True):
64         # Node parent untuk dicari kemungkinan gerakanya
```

```

65     parent = hq.heappop(prioQueue)
66     visited.add(parent.buffer)
67     r = parent.emptyTile[0]
68     c = parent.emptyTile[1]
69     # Kemungkinan gerak ke atas
70     if (r != 0):
71         tempMatrix = move(parent,r,c,r-1,c)
72         if (tempMatrix not in visited):
73             nodes += 1
74             cost = fungsiCost(parent.cost,r,c,tempMatrix[(r*4)+c],r-1,c)
75             tempNode = Node(parent.level+1, cost, tempMatrix, [r-1,c], parent)
76             hq.heappush(prioQueue,tempNode)
77             visited.add(tempMatrix)
78             if (cost == parent.level+1):
79                 return [tempNode, nodes]
80     # Kemungkinan gerak ke kiri
81     if (c != 0):
82         tempMatrix = move(parent,r,c,r,c-1)
83         if (tempMatrix not in visited):
84             nodes += 1
85             cost = fungsiCost(parent.cost,r,c,tempMatrix[(r*4)+c],r,c-1)
86             tempNode = Node(parent.level+1, cost, tempMatrix, [r,c-1], parent)
87             hq.heappush(prioQueue,tempNode)
88             visited.add(tempMatrix)
89             if (cost == parent.level+1):
90                 return [tempNode, nodes]
91     # Kemungkinan gerak ke kanan
92     if (c != 3):
93         tempMatrix = move(parent,r,c,r,c+1)
94         if (tempMatrix not in visited):
95             nodes += 1
96             cost = fungsiCost(parent.cost,r,c,tempMatrix[(r*4)+c],r,c+1)
97             tempNode = Node(parent.level+1, cost, tempMatrix, [r,c+1], parent)
98             hq.heappush(prioQueue, tempNode)
99             visited.add(tempMatrix)
100             if (cost == parent.level+1):
101                 return [tempNode, nodes]
102     # Kemungkinan gerak ke kanan
103     if (r != 3):
104         tempMatrix = move(parent,r,c,r+1,c)
105         if (tempMatrix not in visited):
106             nodes += 1
107             cost = fungsiCost(parent.cost,r,c,tempMatrix[(r*4)+c],r+1,c)
108             tempNode = Node(parent.level+1, cost, tempMatrix, [r+1,c], parent)
109             hq.heappush(prioQueue, tempNode)
110             visited.add(tempMatrix)
111             if (cost == parent.level+1):
112                 return [tempNode, nodes]
113
114     # Fungsi pembacaan file pada folder ../testcase/
115     def readFile(filename):
116         tempList = list([])
117         try:
118             with open("../testcase/" + filename) as f:
119                 content = f.readlines()
120         except:
121             print("File tidak ditemukan")
122             exit()
123         for x in content:
124             i = 0
125             while (i < len(x)) and (x[i] != '\n'):
126                 if (x[i] != ' ') and (x[i+1] == ' '):
127                     tempList.append(int(x[i]))
128                     i += 1

```

```

129         elif (x[i] != ' ') and (x[i+1] != ' '):
130             tempList.append(int(x[i:i+2]))
131             i += 2
132         else:
133             i += 1
134     return tuple(tempList)
135
136
137 if __name__ == "__main__":
138     # Deklarasi variabel
139     prioQueue = []
140     visited = set()
141     puzzle = tuple([])
142     cost = 0
143     r = 0
144     c = 0
145
146     # Membaca file input
147     input = input("Masukkan nama file: ")
148     puzzle = readFile(input)
149
150     tAwal = time.time()
151
152     # Mencari posisi emptyTile
153     for i in range(0,16):
154         if puzzle[i] == 16:
155             r = i // 4
156             c = i % 4
157             break
158
159     # Mencari cost awal
160     for i in range(0,16):
161         if puzzle[i] != 16 and puzzle[i] != i+1:
162             cost += 1
163
164     # Pencarian solusi
165     if fungsiKurang(puzzle) % 2 == 0:
166         hq.heappush(prioQueue, (Node(0, cost, puzzle, [r,c], None)))
167         end = cariKemungkinan(prioQueue, visited)
168         tAkhir = time.time()
169
170     # Menampilkan hasil dan informasinya
171     pathway = []
172     paths = end[0].level
173     print("Jalur terpendek pada puzzle ini adalah:")
174     while (end[0] != None):
175         pathway.append(changeToZero(np.matrix(end[0].buffer)))
176         end[0] = end[0].parent
177     for i in range(len(pathway)-1, -1, -1):
178         print(pathway[i], "\n")
179     print("Nilai kurang:", fungsiKurang(puzzle))
180     print("Jumlah pergerakan terpendek:", paths)
181     print("Waktu:", tAkhir - tAwal, "detik")
182     print("Simpul:", end[1])
183     else:
184         print("Nilai kurang:", fungsiKurang(puzzle))
185         print("Cannot be solved")

```

E. Instansiasi 5 buah persoalan 15-Puzzle

Untuk 5 buah testcase yang telah dibuat terdapat pada folder test pada sourcecode yang dapat di akses di alamat repository di bawah. Untuk hasil dari percobaan kelima testcase tersebut, dapat dilihat di bagian B laporan ini.

F. Alamat repository:

<https://github.com/arikrayi/Tucil-3-Branch-and-Bound>