Tugas Kecil 3 IF2211 Strategi Algoritma
Semester II tahun 2021/2022

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

Dibuat oleh:

Felicia Sutandijo – 13520050

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

A. Algoritma *Branch and Bound*

Berikut adalah algoritma *branch and bound* yang digunakan dalam program pencari solusi 15-puzzle:

1. Program menerima suatu konfigurasi papan permainan 15-puzzle yang masih acak dan mencoba mencari jalan untuk menyelesaikan puzzle tersebut.

```
01 02 03 04
-- 05 06 08
09 10 07 11
13 14 15 12
```

2. Untuk menentukan apakah *puzzle* dapat diselesaikan, perlu ditentukan nilai Σ Kurang(i) + X dari *puzzle* tersebut. Bila nilai Σ Kurang(i) + X dari *puzzle* genap, maka *puzzle* dapat diselesaikan. Sebaliknya, bila nilai tersebut ganjil, *puzzle* tidak dapat diselesaikan dan program berhenti.

Tabel 1. Nilai Kurang(i)

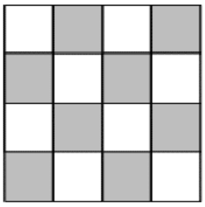
Ubin	**Nilai Kurang(i)**
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1 (ubin 7)
9	1 (ubin 7)
10	1 (ubin 7)
11	0
12	0
13	1 (ubin 12)
14	1 (ubin 12)
15	1 (ubin 12)
ubin kosong (16)	11 (ubin 5, 6, 8, 9, 10, 7, 11, 12, 14, 15, 12)
Total	**17**

KURANG(i) = banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i).

POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa.

X = 1 bila sel kosong pada posisi awal berada pada sel yg diarsir.

X = 0 bila sel kosong pada posisi awal berada pada sel yg tidak diarsir.



Σ Kurang(i) = 17

X = 1

Σ Kurang(i) + X = 17 + 1 = 18

Σ Kurang(i) + X **genap**.

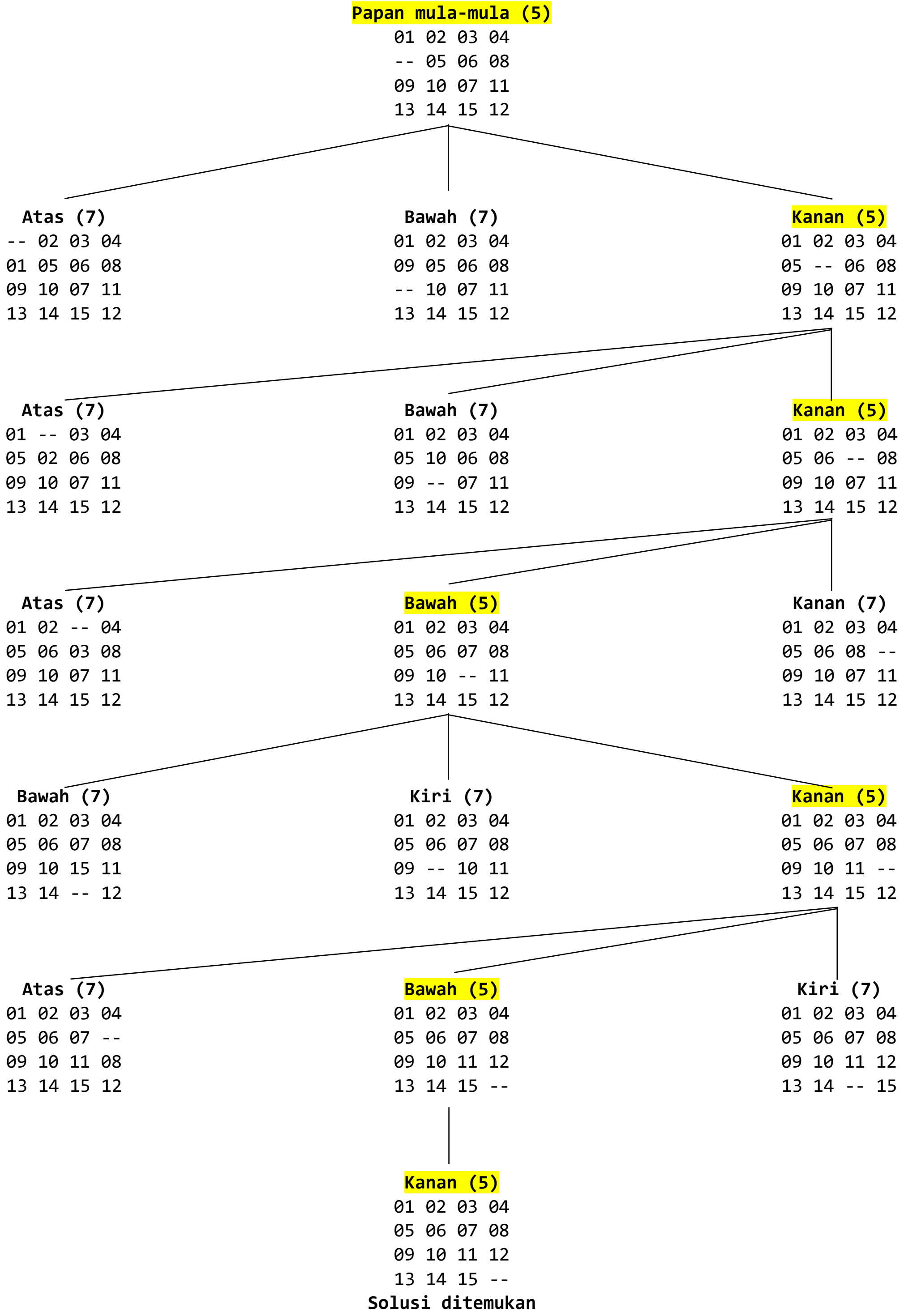
Kesimpulan: ***puzzle* dapat diselesaikan.**

3. Bila *puzzle* dapat diselesaikan, algoritma lanjut ke pencarian solusi menggunakan algoritma *branch and bound* dengan *cost* setiap *node* dihitung dengan menjumlahkan langkah yang telah diambil untuk mencapai *node* tersebut dengan *cost* perkiraan *node* untuk mencapai *goal*. Pada algoritma ini, *cost* perkiraan tersebut merupakan berapa ubin tidak kosong yang masih belum di posisi yang benar. Secara matematika, hal ini dapat dituliskan sebagai berikut.

c(i) = f(i) + g(i)
c(i) = ongkos untuk simpul i
f(i) = ongkos mencapai simpul i dari akar = langkah yang telah ditempuh untuk mencapai simpul i
g(i) = perkiraan ongkos mencapai simpul tujuan dari simpul i = jumlah ubin yang belum pada tempatnya

```
               Papan mula-mula
                 01 02 03 04
                 -- 05 06 08
                 09 10 07 11
                 13 14 15 12

        Langkah yang telah dilakukan: 0
Jumlah ubin yang tidak pada tempatnya: 5 (Ubin 05, 06, 07, 11, 12)
               Cost: 0 + 5 = 5
```

4. Implementasi algoritma *branch and bound* menggunakan sebuah *priority queue* untuk mencatat simpul-simpul hidup dan sebuah *dictionary* untuk mencatat simpul-simpul yang telah dikunjungi (agar tidak dikunjungi dua kali).



```
                       Papan mula-mula (5)
                           01 02 03 04
                           -- 05 06 08
                           09 10 07 11
                           13 14 15 12

   Atas (7)                 Bawah (7)                 Kanan (5)
 -- 02 03 04               01 02 03 04               01 02 03 04
 01 05 06 08               09 05 06 08               05 -- 06 08
 09 10 07 11               -- 10 07 11               09 10 07 11
 13 14 15 12               13 14 15 12               13 14 15 12

   Atas (7)                 Bawah (7)                 Kanan (5)
 01 -- 03 04               01 02 03 04               01 02 03 04
 05 02 06 08               05 10 06 08               05 06 -- 08
 09 10 07 11               09 -- 07 11               09 10 07 11
 13 14 15 12               13 14 15 12               13 14 15 12

   Atas (7)                 Bawah (5)                 Kanan (7)
 01 02 -- 04               01 02 03 04               01 02 03 04
 05 06 03 08               05 06 07 08               05 06 08 --
 09 10 07 11               09 10 -- 11               09 10 07 11
 13 14 15 12               13 14 15 12               13 14 15 12

  Bawah (7)                 Kiri (7)                  Kanan (5)
 01 02 03 04               01 02 03 04               01 02 03 04
 05 06 07 08               05 06 07 08               05 06 07 08
 09 10 15 11               09 -- 10 11               09 10 11 --
 13 14 -- 12               13 14 15 12               13 14 15 12

   Atas (7)                 Bawah (5)                 Kiri (7)
 01 02 03 04               01 02 03 04               01 02 03 04
 05 06 07 --               05 06 07 08               05 06 07 08
 09 10 11 08               09 10 11 12               09 10 11 12
 13 14 15 12               13 14 15 --               13 14 -- 15

                            Kanan (5)
                           01 02 03 04
                           05 06 07 08
                           09 10 11 12
                           13 14 15 --
                        Solusi ditemukan
```

5. Algoritma berjalan terus hingga solusi ditemukan atau seluruh simpul hidup dalam *priority queue* telah dikunjungi.

6. Solusi dicetak ke layer beserta waktu yang dibutuhkan program serta jumlah simpul hidup yang telah dibangkitkan. Jumlah simpul yang dibangkitkan tidak termasuk simpul akar dan simpul solusi.

```
Berhasil diselesaikan!
Awalnya gini:
01 02 03 04
-- 05 06 08
09 10 07 11
13 14 15 12

Langkah ke-1:
01 02 03 04
05 -- 06 08
09 10 07 11
13 14 15 12

Langkah ke-2:
01 02 03 04
05 06 -- 08
09 10 07 11
13 14 15 12

Langkah ke-3:
01 02 03 04
05 06 07 08
09 10 -- 11
13 14 15 12

Langkah ke-4:
01 02 03 04
05 06 07 08
09 10 11 --
13 14 15 12

Langkah ke-5:
01 02 03 04
05 06 07 08
09 10 11 12
13 14 15 --

Langkah yang diperlukan: 5
Waktu yang diperlukan: 0.0052 detik
Jumlah simpul yang dibangkitkan: 15
```

B. *Source Code* Program dalam Python

1. Program Utama (main.py)

```python
from boardGenerator import randomBoard
from reader import read
from reachableGoal import *
from branchAndBound import branchAndBound
import sys
cyan = "\033[96m"
red = "\033[91m"
end = "\033[0m"

# Read file if given, otherwise use a random board
if (len(sys.argv) > 1):
  fileName = sys.argv[1]
  try:
    board = read(fileName)
  except FileNotFoundError:
    print("File tidak ditemukan.")
    sys.exit(1)
else:
  board = randomBoard()

# Output initial board
print("Susunan awal:")
board.print()

# Output kurang(i) of the initial board
timeElapsedKurang_i, kurang = kurang_i(board)
print("Nilai Kurang(i):")
for key in sorted(kurang):
  print(f"{key}: {kurang[key]}")
print()

# Output the sum of all values in kurang(i) + X
sums = sumAll(kurang, X(board))
print(f"Nilai sigma Kurang(i) + X: {sums}\n")

# Perform branch and bound if the board is solvable, otherwise print an error message
timeElapsedBnB = 0
nodeCount = 0
if not reachableGoal(sums):
  print(f"{red}Persoalan tidak dapat diselesaikan.\n{end}")
else:
  timeElapsedBnB, nodeCount = branchAndBound(board)

# Output time taken and number of nodes generated
```

```
print(cyan, end="")
print(f"Waktu yang diperlukan: {timeElapsedKurang_i+timeElapsedBnB:0.4f}
detik")
print(f"Jumlah simpul yang dibangkitkan: {nodeCount}")
print(end, end="")
```

2. Kelas Board (board.py)

```
class Board:
  # A class that represents a board of 15-puzzle

  # Attributes:
  #   blocks: an array of integers representing the blocks on the board
  #   steps: an integer representing the number of steps taken to reach the
board
  #   prevStep: a Board object representing the previous step taken to
reach the board
  #   misplacedBlocks: an integer representing the number of misplaced
blocks on the board

  # Methods:
  #   copy: a function that returns a copy of the board and adds one step
  #   print: a function that prints the board
  #   emptyBlock: a function that returns the index of the empty block on
the board

  #   swap: a function that returns a copy of the board with the blocks
swapped
  #   up: a function that returns a copy of the board with the empty block
swapped with the block above it
  #   down: a function that returns a copy of the board with the empty
block swapped with the block below it
  #   left: a function that returns a copy of the board with the empty
block swapped with the block to the left of it
  #   right: a function that returns a copy of the board with the empty
block swapped with the block to the right of it

  #   countMisplacedBlocks: a function that returns the number of misplaced
blocks on the board
  #   isGoal: a function that returns True if the board is a goal board,
False otherwise
  #   cost: a function that returns the cost of the board
  #   __lt__: a function that returns True if the board has a lower cost
than the other board, False otherwise

  def __init__(self, blocks, steps):
    self.steps = steps
    self.blocks = blocks
    self.prevStep = None
    self.misplacedBlocks = self.countMisplacedBlocks()

  def copy(self):
    return Board(self.blocks[:], self.steps+1)

  def print(self):
```

```
  for i in range(4):
    for j in range(4):
      if (self.blocks[i*4+j] < 10):
        print(f"0{self.blocks[i*4+j]}", end=" ")
      elif (self.blocks[i*4+j] == 16):
        print("--", end=" ")
      else:
        print(self.blocks[i*4+j], end=" ")
    print()
  print()

def emptyBlock(self):
  for i in range(16):
    if self.blocks[i] == 16:
      return i

def swap(self, i, j):
  result = self.copy()
  result.prevStep = self
  temp = result.blocks[i]
  result.blocks[i] = result.blocks[j]
  result.blocks[j] = temp
  result.misplacedBlocks = result.countMisplacedBlocks()
  return result

def up(self):
  if self.emptyBlock() < 4:
    return None
  else:
    up = self.swap(self.emptyBlock()-4, self.emptyBlock())
    return up

def down(self):
  if self.emptyBlock() > 11:
    return None
  else:
    down = self.swap(self.emptyBlock()+4, self.emptyBlock())
    return down

def left(self):
  if self.emptyBlock() % 4 == 0:
    return None
  else:
    left = self.swap(self.emptyBlock()-1, self.emptyBlock())
    return left

def right(self):
  if self.emptyBlock() % 4 == 3:
    return None
  else:
    right = self.swap(self.emptyBlock()+1, self.emptyBlock())
    return right

def countMisplacedBlocks(self):
  misplaced = 0
  for i in range(16):
    if self.blocks[i] != 16 and self.blocks[i] != i+1:
      misplaced += 1
  return misplaced

def isGoal(self):
  return self.misplacedBlocks == 0

def cost(self):
  return self.steps+self.misplacedBlocks

def __lt__(self, other):
  if(self.steps<other.steps):
    return self
  else:
    return other
```

3. Pemroses File ke Board (reader.py)

```
from board import Board

def read(fileName):
  # A function that reads a board from a file and returns a Board object

  # Read file
  with open(fileName, "r") as f:
    data = f.read()

  board = [] # Initiating the board

  # Processing data
  data = data.split("\n")
  for i in range(4):
    data[i] = data[i].split(" ")
    for j in range(4):
      curr = data[i][j]
   if curr == "-" or curr == "0" or curr == "--": # Empty block is
converted to 16
        board.append(16)
      else:
        board.append(int(curr))

  # Creating a Board object
  return Board(board, 0)
```

4. Pembangkit Papan Permainan Acak 15-Puzzle (randomBoard.py)

```python
import random
from board import Board

def randomBoard():
  # A function that returns a random board of 15-puzzle
  blocks = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] # Initiating the blocks
  random.shuffle(blocks) # Shuffling the blocks
  board = Board(blocks, 0) # Creating a Board object
  return board
```

5. Kurang(i) dan X (reachableGoal.py)

```
from time import perf_counter

def kurang_i(board):
  # A function that returns the number of blocks that are misplaced
  # The board is represented as an array of strings
  # The function will return a dictionary
  timeStart = perf_counter() # Start the timer
  board_copy = board.blocks[:] # Make a copy of the board blocks
configuration
  kurang = {} # Initiate the dictionary
  for i in range(16):
    curr = (board_copy[i])
    kurang[curr] = 0
    for j in range(i+1, 16):
      if curr > (board_copy[j]):
        kurang[curr] += 1
  timeEnd = perf_counter() # End the timer
  return timeEnd-timeStart, kurang

def X(board):
  # Function X(board) receives a board
  # and returns the value of X (0 or 1)
  idx = board.emptyBlock() # find the index of the empty block
  return ((idx//4 + idx%4)%2) # 0 if row+col is even, 1 if row+col is odd

def sumAll(kurang_i, X):
  # Function sumAll(kurang_i, X) receives the value of kurang_i and X
  # and returns the sum of all values in kurang_i + X
  sumOfKurang_i = sum(kurang_i.values())
  return (sumOfKurang_i + X)

def reachableGoal(sumAll):
  # Function reachableGoal(sumAll) receives the value of the sum of all
values in kurang_i + X
  # and returns True if it is even, False otherwise
  return ((sumAll)%2 == 0)
```

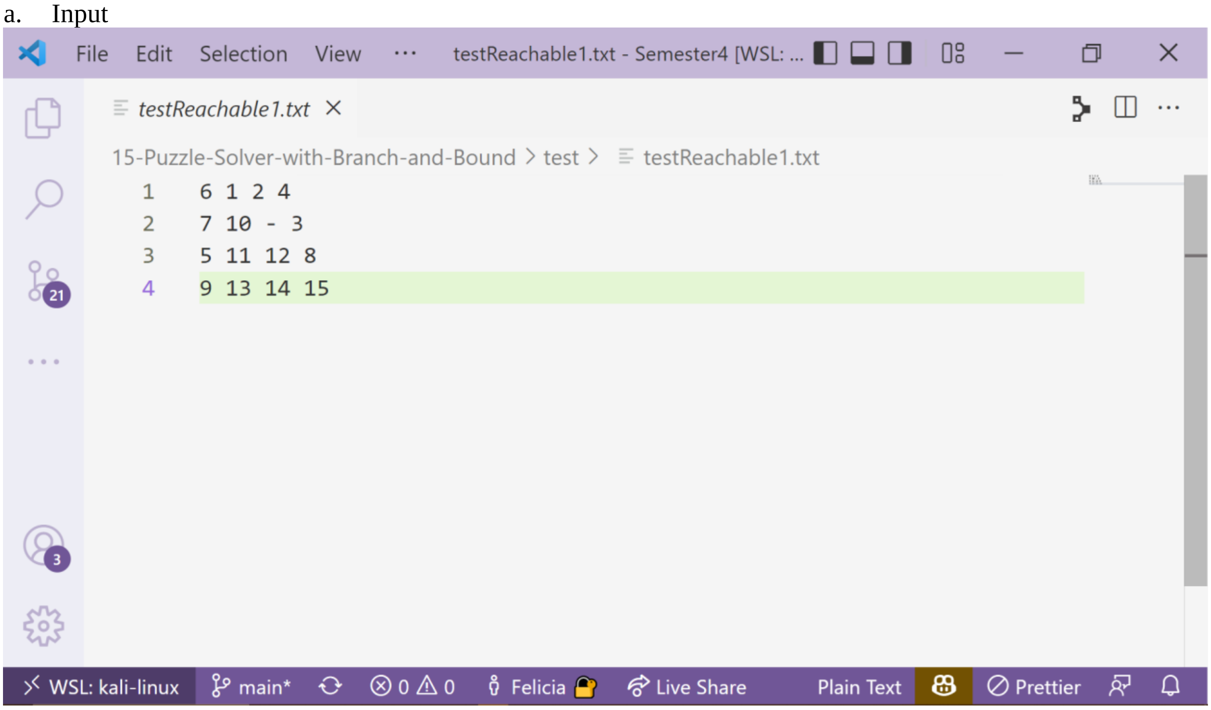
6. Algoritma Branch and Bound (branchAndBound.py)

```
from queue import PriorityQueue
from time import perf_counter

def branchAndBound(start):
  # Function branchAndBound accepts initial board,
  # prints the steps taken to reach goal,
  # and returns the time taken and number of nodes generated.

  # Formats ehe, don't mind this :D
  green = "\033[92m"
  cyan = "\033[96m"
  red = "\033[91m"
  u = "\033[4m"
  b = "\033[1m"
  end = "\033[0m"

  timeStart = perf_counter() # Start timer

  # Initialize variables
  nodeCount = 0 # Number of nodes generated
  liveNodes = PriorityQueue() # PriorityQueue for live nodes
  liveNodes.put((start.cost(), start))
  curr = start
  checked = {str(curr.blocks) : True} # Dictionary for checked nodes

  # Iterate through liveNodes until goal is reached or all nodes are checked
  while (not curr.isGoal()) and (not liveNodes.empty()):
    # Take the node with the lowest cost from liveNodes
    curr = liveNodes.get()[1]
    # Add current node to checked nodes
    checked[str(curr.blocks)] = True

    # For each node, try to swap empty block up, down, left, and right
    # and add the swapped board to liveNodes if it's not yet checked
    up = curr.up()
    if (up is not None) and (str(up.blocks) not in checked):
      liveNodes.put((up.cost(), up))
      checked[str(up.blocks)] = True
      nodeCount += 1

    down = curr.down()
    if (down is not None) and (str(down.blocks) not in checked):
      liveNodes.put((down.cost(), down))
      checked[str(down.blocks)] = True
      nodeCount += 1

    left = curr.left()
  if (left is not None) and (str(left.blocks) not in checked):
    liveNodes.put((left.cost(), left))
    checked[str(left.blocks)] = True
    nodeCount += 1

  right = curr.right()
  if (right is not None) and (str(right.blocks) not in checked):
    liveNodes.put((right.cost(), right))
    checked[str(right.blocks)] = True
    nodeCount += 1
# Goal is reached

timeEnd = perf_counter() # End timer

# Output steps
if (curr.steps == 0):
  print(f"{red}{b}Loh puzzle-nya sudah selesai. >:(\n{end}{end}")
else:
  print(f"{green}{b}Berhasil diselesaikan!{end}{end}")
  stepsToSuccess = []
  currStep = curr

  while (currStep.prevStep is not None):
    stepsToSuccess.append(currStep)
    currStep = currStep.prevStep

  print(f"{u}Awalnya gini:{end}")
  start.print()

  for i in range(len(stepsToSuccess)-1, -1, -1):
    print(f"{u}Langkah ke-{len(stepsToSuccess)-i}:{end}")
    stepsToSuccess[i].print()

  print(f"{cyan}Langkah yang diperlukan: {curr.steps}{end}")

return (timeEnd-timeStart, nodeCount)
```

C. *Screenshot* Input dan Output

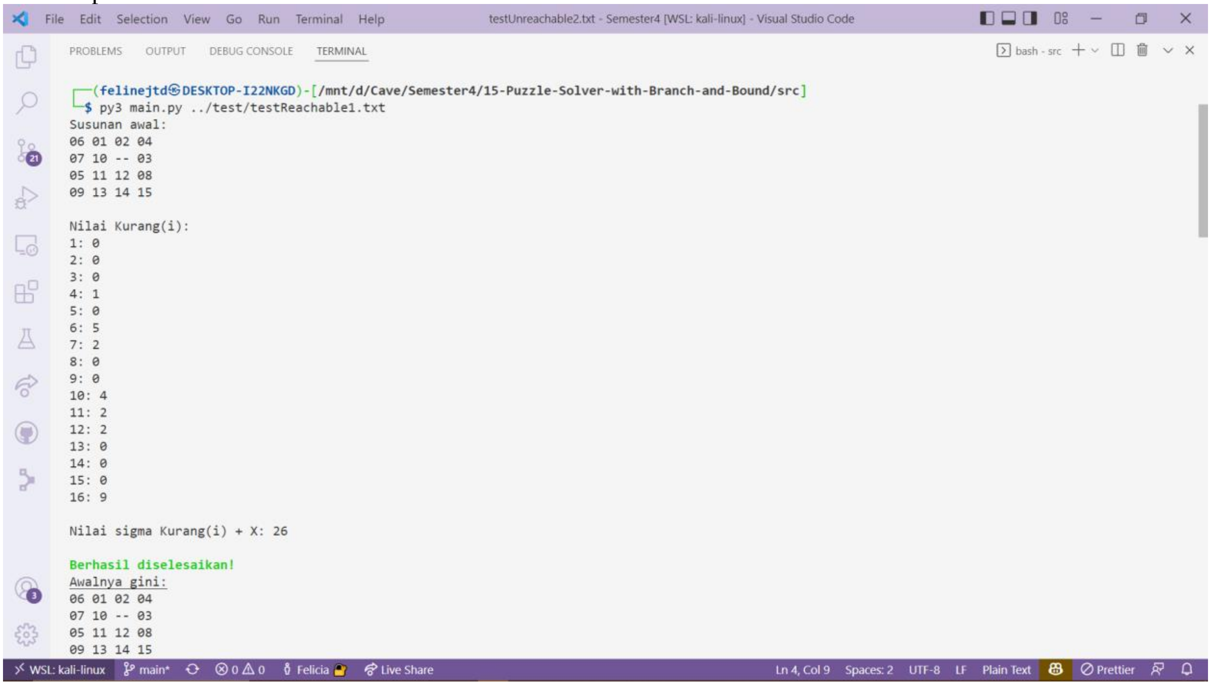
1. Test Case Reachable 1

a. Input

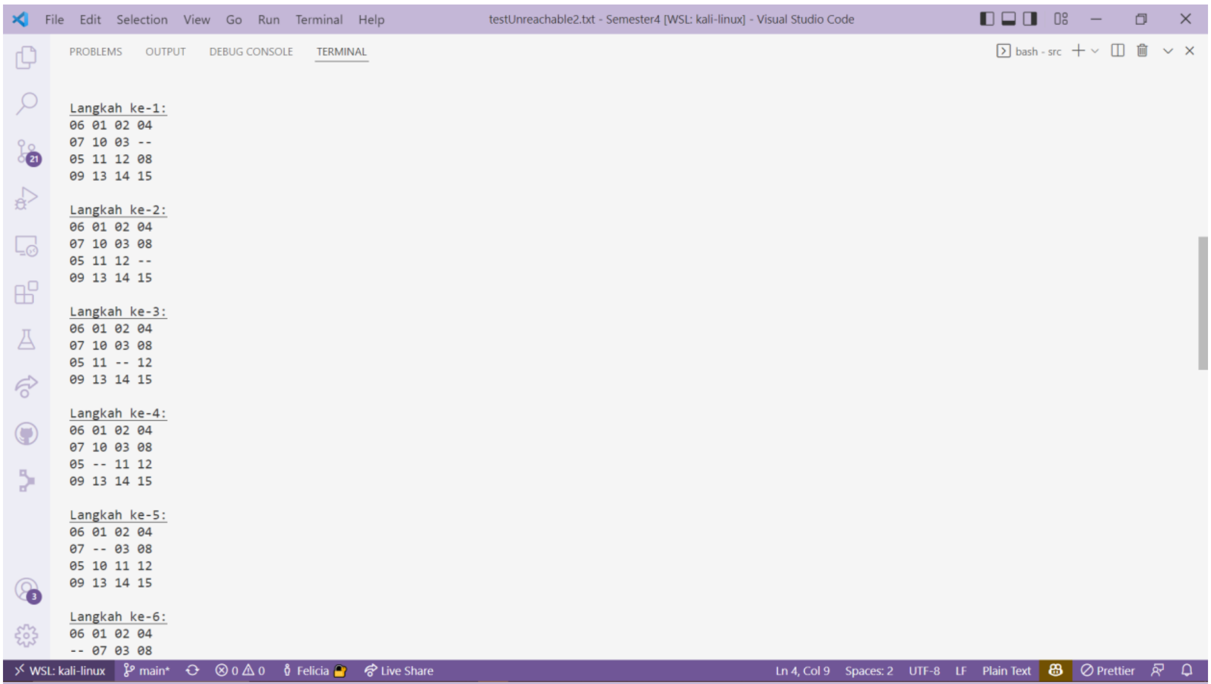

File
Edit
Selection
View
testReachable1.txt - Semester4 [WSL: ...
testReachable1.txt
15-Puzzle-Solver-with-Branch-and-Bound > test > testReachable1.txt
WSL: kali-linux
main*
0 0
Felicia
Live Share
Plain Text
Prettier

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

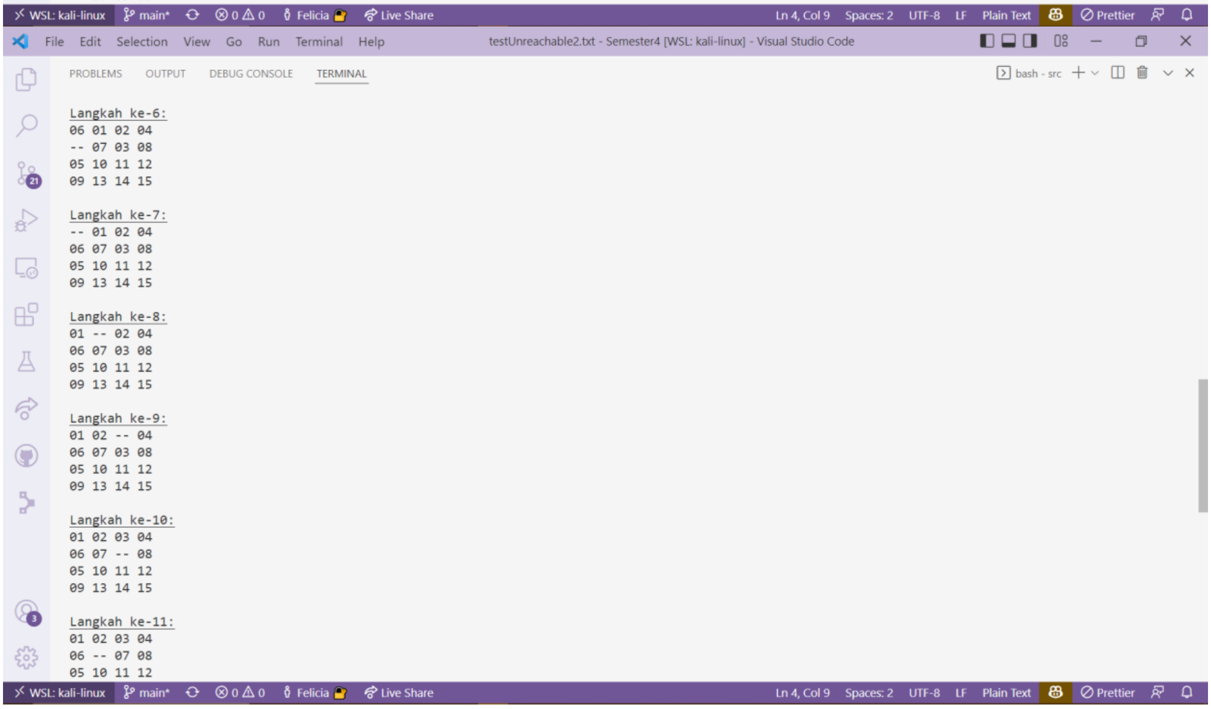
b. Output



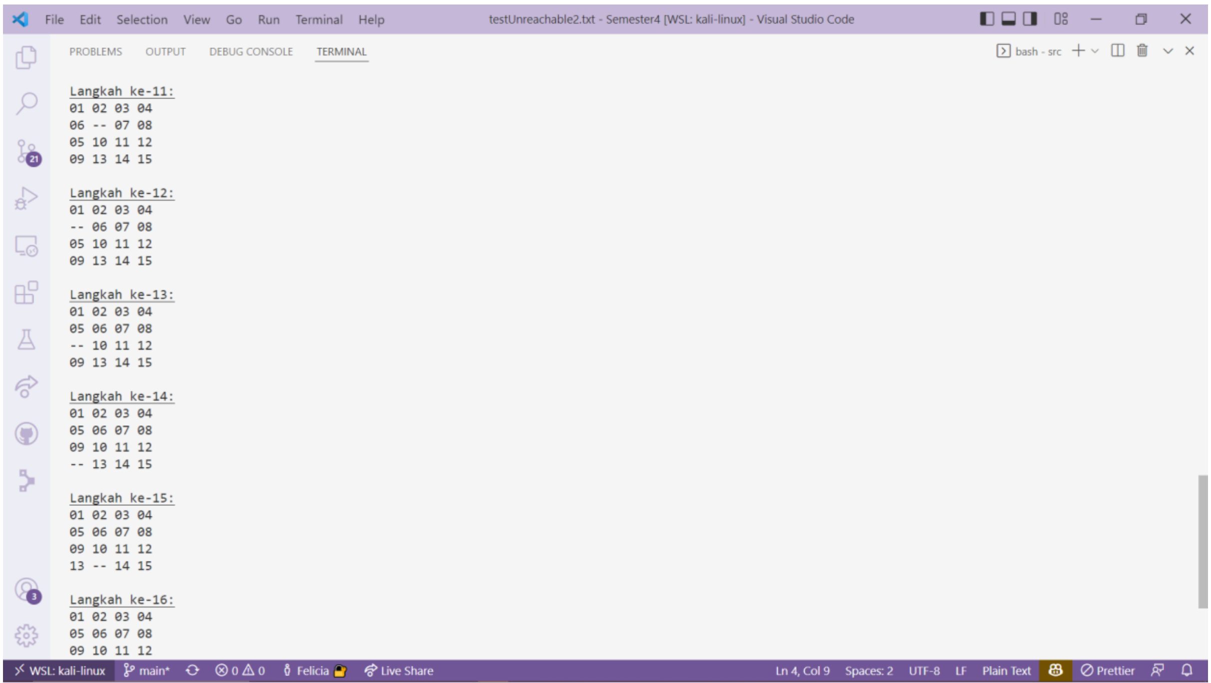
```
┌──(felinejtd㉿DESKTOP-I22NKGD)-[/mnt/d/Cave/Semester4/15-Puzzle-Solver-with-Branch-and-Bound/src]
└─$ py3 main.py ../test/testReachable1.txt
Susunan awal:
06 01 02 04
07 10 -- 03
05 11 12 08
09 13 14 15

Nilai Kurang(i):
1: 0
2: 0
3: 0
4: 1
5: 0
6: 5
7: 2
8: 0
9: 0
10: 4
11: 2
12: 2
13: 0
14: 0
15: 0
16: 9

Nilai sigma Kurang(i) + X: 26

Berhasil diselesaikan!
Awalnya gini:
06 01 02 04
07 10 -- 03
05 11 12 08
09 13 14 15
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux
main*
0 0
Felicia
Live Share
Ln 4, Col 9
Spaces: 2
UTF-8
LF
Plain Text
Prettier

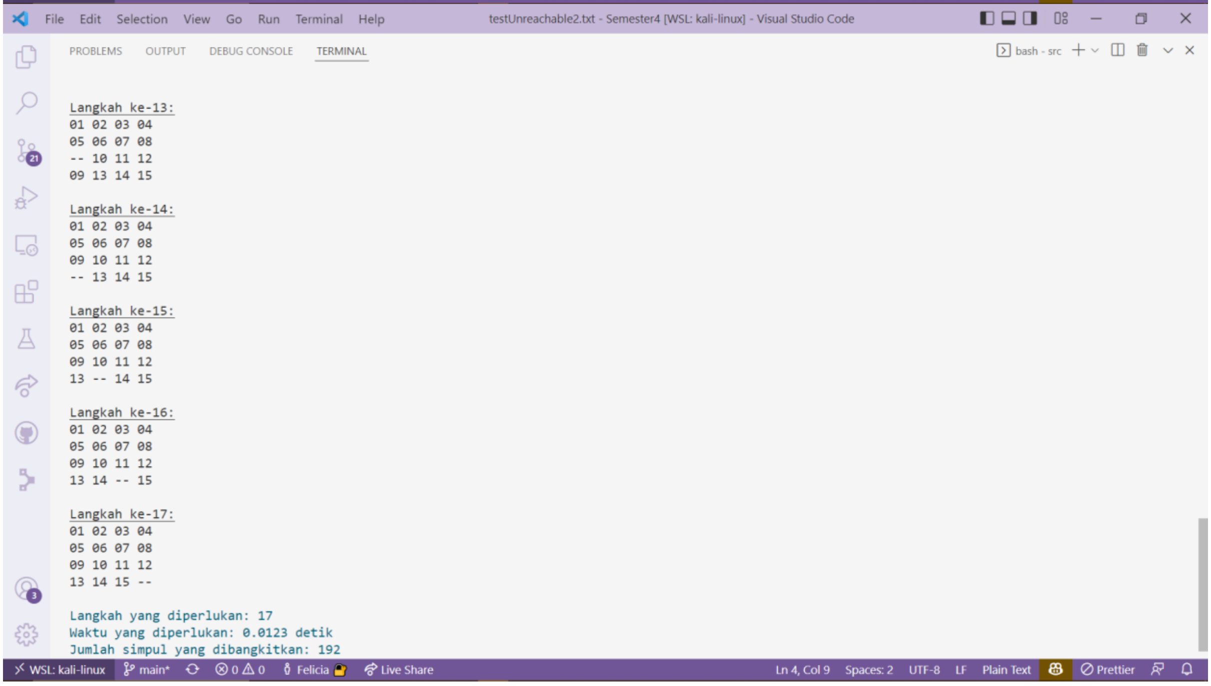
```
Langkah ke-1:
06 01 02 04
07 10 03 --
05 11 12 08
09 13 14 15

Langkah ke-2:
06 01 02 04
07 10 03 08
05 11 12 --
09 13 14 15

Langkah ke-3:
06 01 02 04
07 10 03 08
05 11 -- 12
09 13 14 15

Langkah ke-4:
06 01 02 04
07 10 03 08
05 -- 11 12
09 13 14 15

Langkah ke-5:
06 01 02 04
07 -- 03 08
05 10 11 12
09 13 14 15

Langkah ke-6:
06 01 02 04
-- 07 03 08
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux
main*
0 0
Felicia
Live Share
Ln 4, Col 9
Spaces: 2
UTF-8
LF
Plain Text
Prettier

```
Langkah ke-6:
06 01 02 04
-- 07 03 08
05 10 11 12
09 13 14 15

Langkah ke-7:
-- 01 02 04
06 07 03 08
05 10 11 12
09 13 14 15

Langkah ke-8:
01 -- 02 04
06 07 03 08
05 10 11 12
09 13 14 15

Langkah ke-9:
01 02 -- 04
06 07 03 08
05 10 11 12
09 13 14 15

Langkah ke-10:
01 02 03 04
06 07 -- 08
05 10 11 12
09 13 14 15

Langkah ke-11:
01 02 03 04
06 -- 07 08
05 10 11 12
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux
main*
Felicia
Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text
Prettier

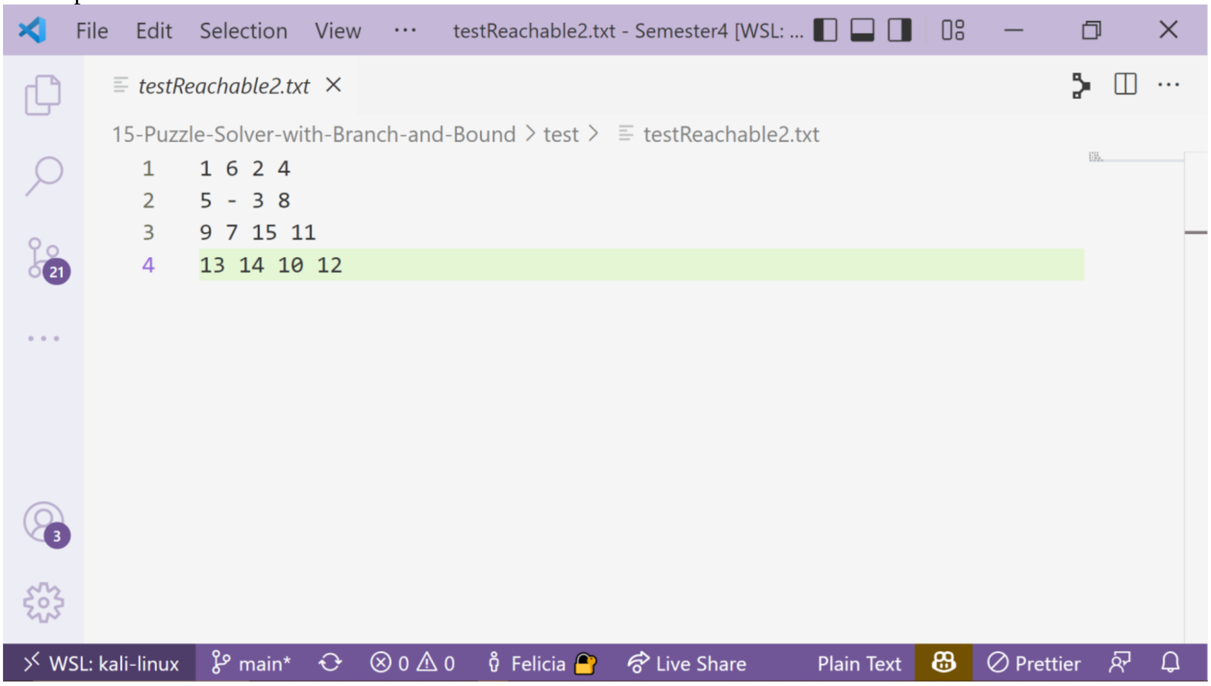
```
Langkah ke-11:
01 02 03 04
06 -- 07 08
05 10 11 12
09 13 14 15

Langkah ke-12:
01 02 03 04
-- 06 07 08
05 10 11 12
09 13 14 15

Langkah ke-13:
01 02 03 04
05 06 07 08
-- 10 11 12
09 13 14 15

Langkah ke-14:
01 02 03 04
05 06 07 08
09 10 11 12
-- 13 14 15

Langkah ke-15:
01 02 03 04
05 06 07 08
09 10 11 12
13 -- 14 15

Langkah ke-16:
01 02 03 04
05 06 07 08
09 10 11 12
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux
main*
Felicia
Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text
Prettier

```
Langkah ke-13:
01 02 03 04
05 06 07 08
-- 10 11 12
09 13 14 15

Langkah ke-14:
01 02 03 04
05 06 07 08
09 10 11 12
-- 13 14 15

Langkah ke-15:
01 02 03 04
05 06 07 08
09 10 11 12
13 -- 14 15

Langkah ke-16:
01 02 03 04
05 06 07 08
09 10 11 12
13 14 -- 15

Langkah ke-17:
01 02 03 04
05 06 07 08
09 10 11 12
13 14 15 --

Langkah yang diperlukan: 17
Waktu yang diperlukan: 0.0123 detik
Jumlah simpul yang dibangkitkan: 192
```

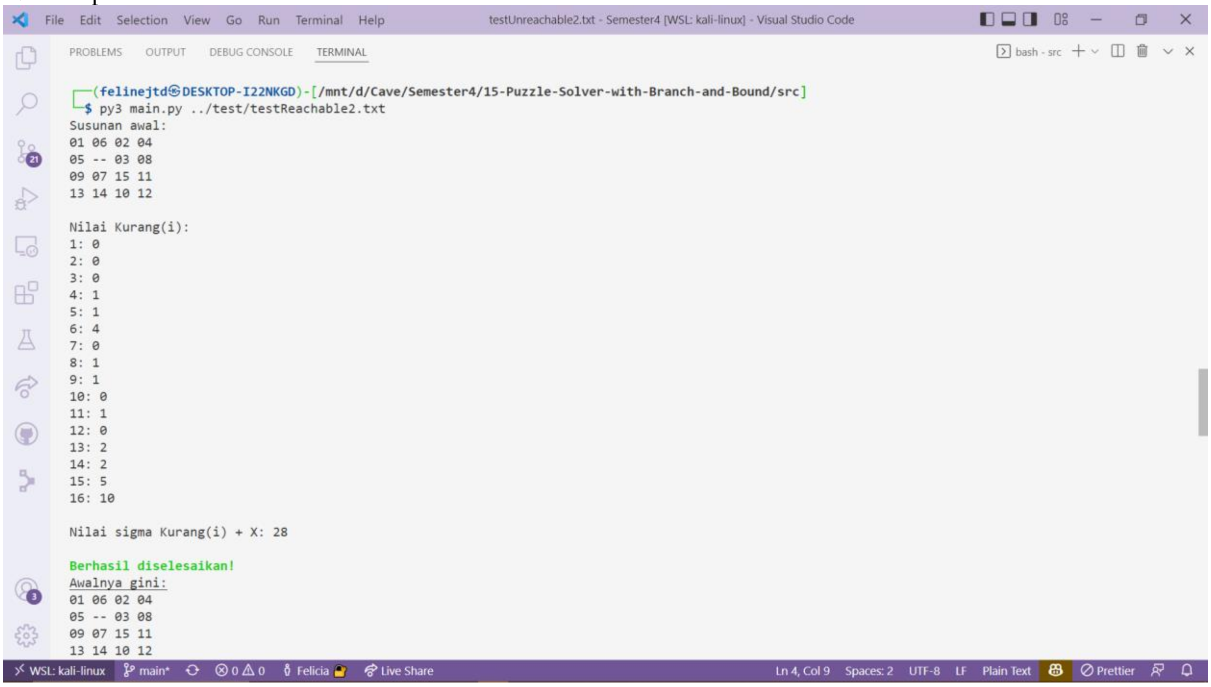
2. Test Case Reachable 2

a. Input

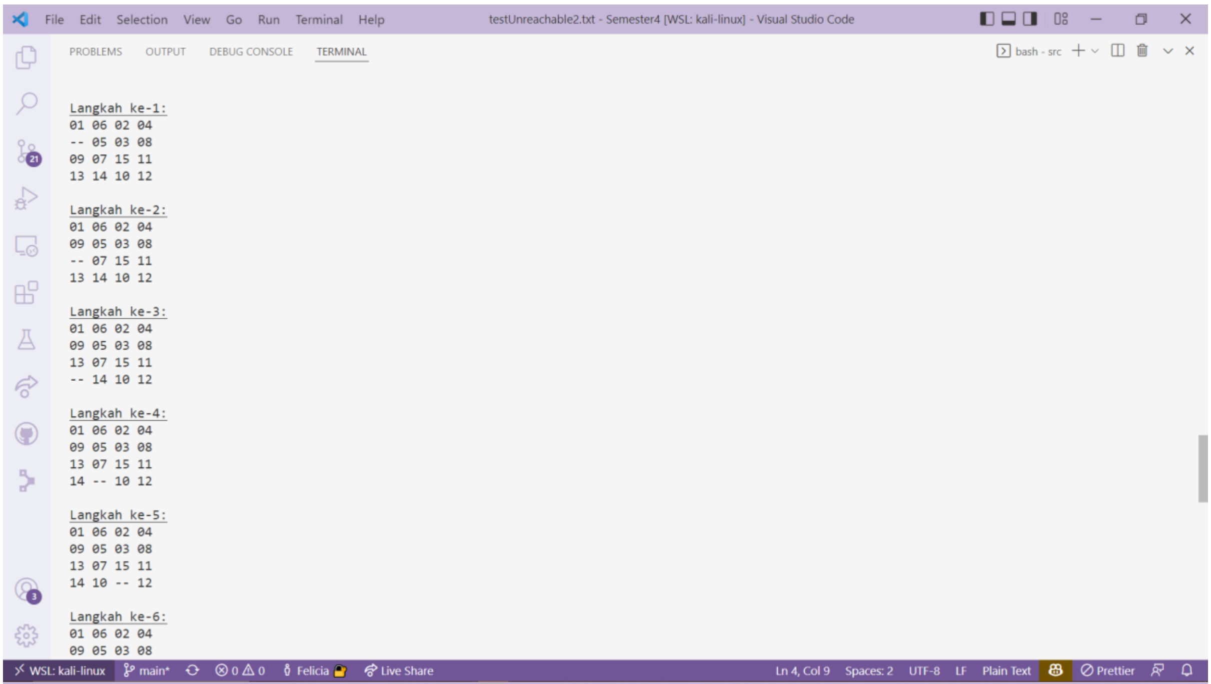

File
Edit
Selection
View
testReachable2.txt - Semester4 [WSL: ...
testReachable2.txt
15-Puzzle-Solver-with-Branch-and-Bound > test > testReachable2.txt
WSL: kali-linux
main*
0 0
Felicia
Live Share
Plain Text
Prettier

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

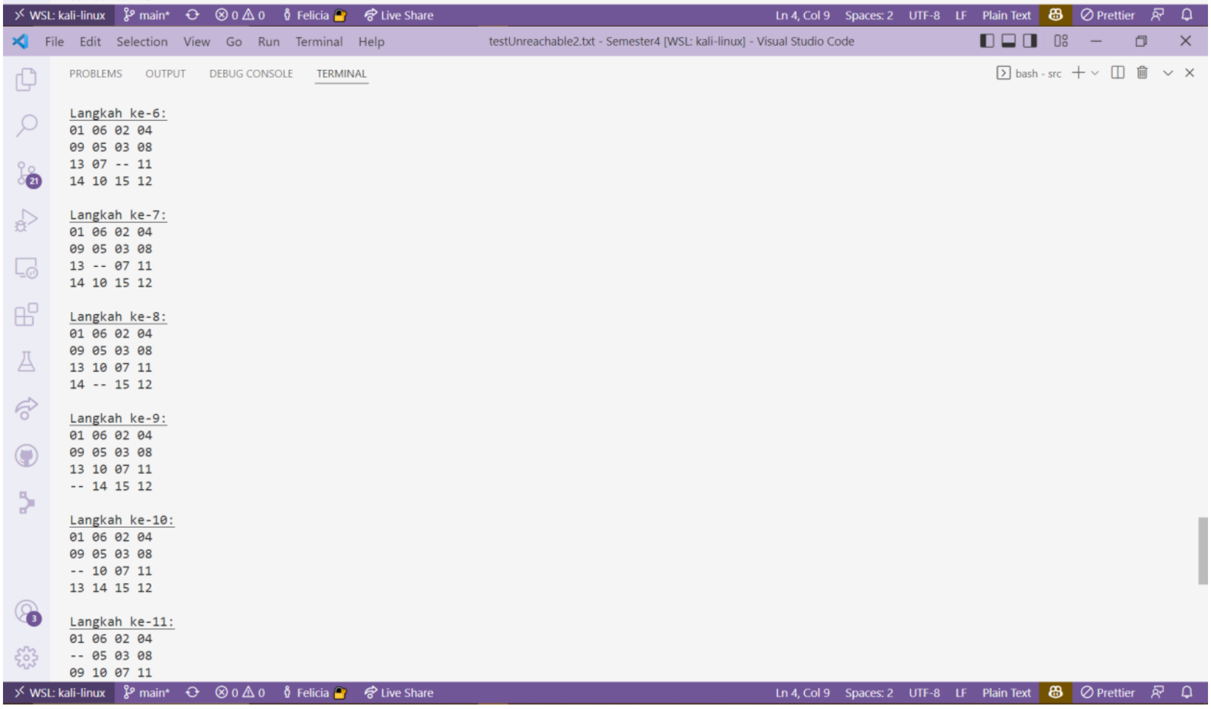
b. Output



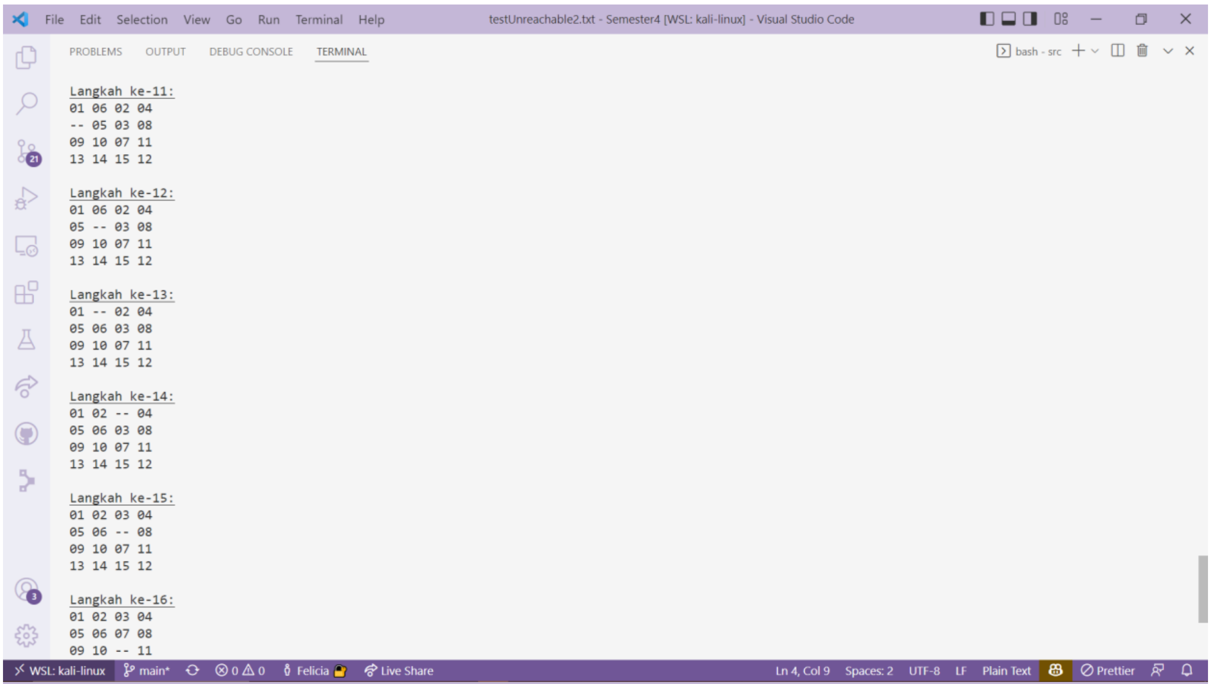
```
┌──(felinejtd㉿DESKTOP-I22NKGD)-[/mnt/d/Cave/Semester4/15-Puzzle-Solver-with-Branch-and-Bound/src]
└─$ py3 main.py ../test/testReachable2.txt
Susunan awal:
01 06 02 04
05 -- 03 08
09 07 15 11
13 14 10 12

Nilai Kurang(i):
1: 0
2: 0
3: 0
4: 1
5: 1
6: 4
7: 0
8: 1
9: 1
10: 0
11: 1
12: 0
13: 2
14: 2
15: 5
16: 10

Nilai sigma Kurang(i) + X: 28

Berhasil diselesaikan!
Awalnya gini:
01 06 02 04
05 -- 03 08
09 07 15 11
13 14 10 12
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux main* 0 0 Felicia Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text Prettier

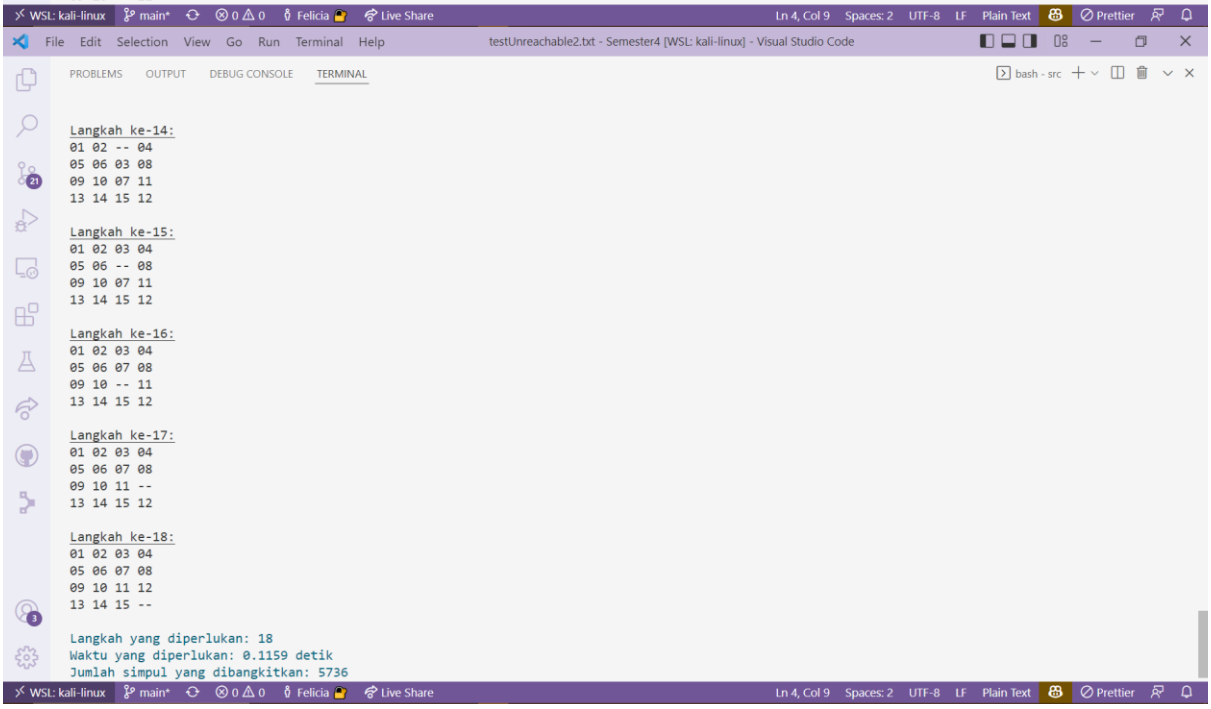
```
Langkah ke-1:
01 06 02 04
-- 05 03 08
09 07 15 11
13 14 10 12

Langkah ke-2:
01 06 02 04
09 05 03 08
-- 07 15 11
13 14 10 12

Langkah ke-3:
01 06 02 04
09 05 03 08
13 07 15 11
-- 14 10 12

Langkah ke-4:
01 06 02 04
09 05 03 08
13 07 15 11
14 -- 10 12

Langkah ke-5:
01 06 02 04
09 05 03 08
13 07 15 11
14 10 -- 12

Langkah ke-6:
01 06 02 04
09 05 03 08
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux main* 0 0 Felicia Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text Prettier

```
Langkah ke-6:
01 06 02 04
09 05 03 08
13 07 -- 11
14 10 15 12

Langkah ke-7:
01 06 02 04
09 05 03 08
13 -- 07 11
14 10 15 12

Langkah ke-8:
01 06 02 04
09 05 03 08
13 10 07 11
14 -- 15 12

Langkah ke-9:
01 06 02 04
09 05 03 08
13 10 07 11
-- 14 15 12

Langkah ke-10:
01 06 02 04
09 05 03 08
-- 10 07 11
13 14 15 12

Langkah ke-11:
01 06 02 04
-- 05 03 08
09 10 07 11
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux
main*
Felicia
Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text
Prettier

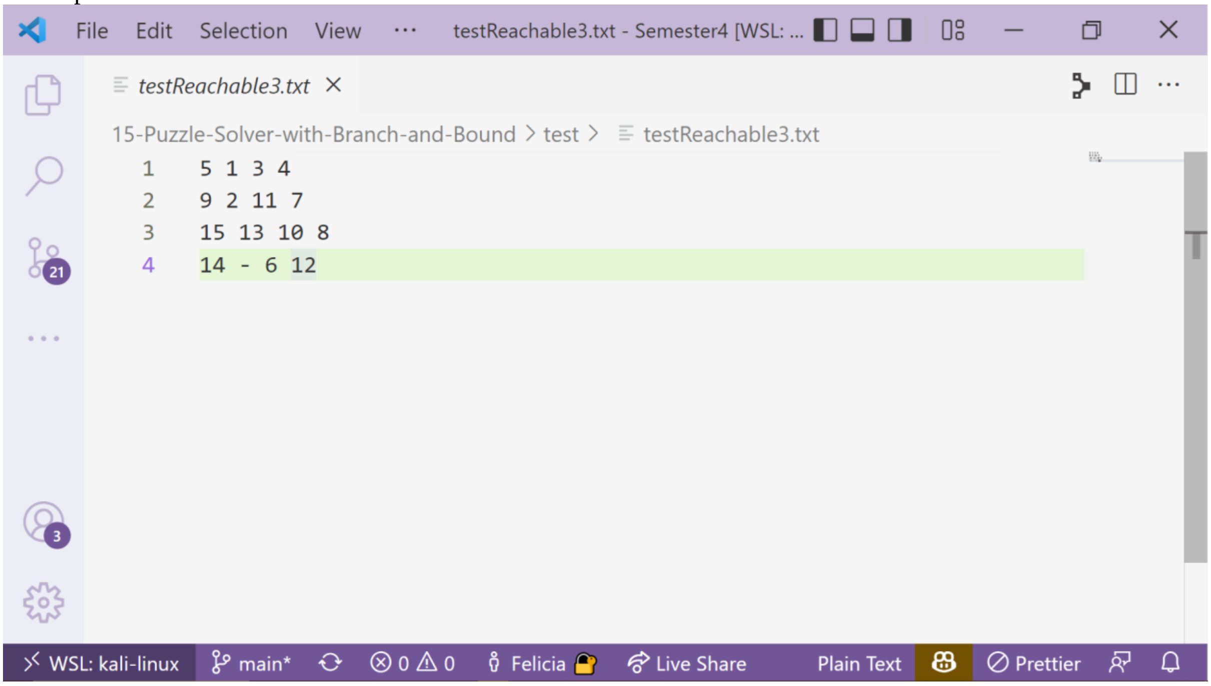
```
Langkah ke-11:
01 06 02 04
-- 05 03 08
09 10 07 11
13 14 15 12

Langkah ke-12:
01 06 02 04
05 -- 03 08
09 10 07 11
13 14 15 12

Langkah ke-13:
01 -- 02 04
05 06 03 08
09 10 07 11
13 14 15 12

Langkah ke-14:
01 02 -- 04
05 06 03 08
09 10 07 11
13 14 15 12

Langkah ke-15:
01 02 03 04
05 06 -- 08
09 10 07 11
13 14 15 12

Langkah ke-16:
01 02 03 04
05 06 07 08
09 10 -- 11
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux
main*
Felicia
Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text
Prettier

```
Langkah ke-14:
01 02 -- 04
05 06 03 08
09 10 07 11
13 14 15 12

Langkah ke-15:
01 02 03 04
05 06 -- 08
09 10 07 11
13 14 15 12

Langkah ke-16:
01 02 03 04
05 06 07 08
09 10 -- 11
13 14 15 12

Langkah ke-17:
01 02 03 04
05 06 07 08
09 10 11 --
13 14 15 12

Langkah ke-18:
01 02 03 04
05 06 07 08
09 10 11 12
13 14 15 --

Langkah yang diperlukan: 18
Waktu yang diperlukan: 0.1159 detik
Jumlah simpul yang dibangkitkan: 5736
```

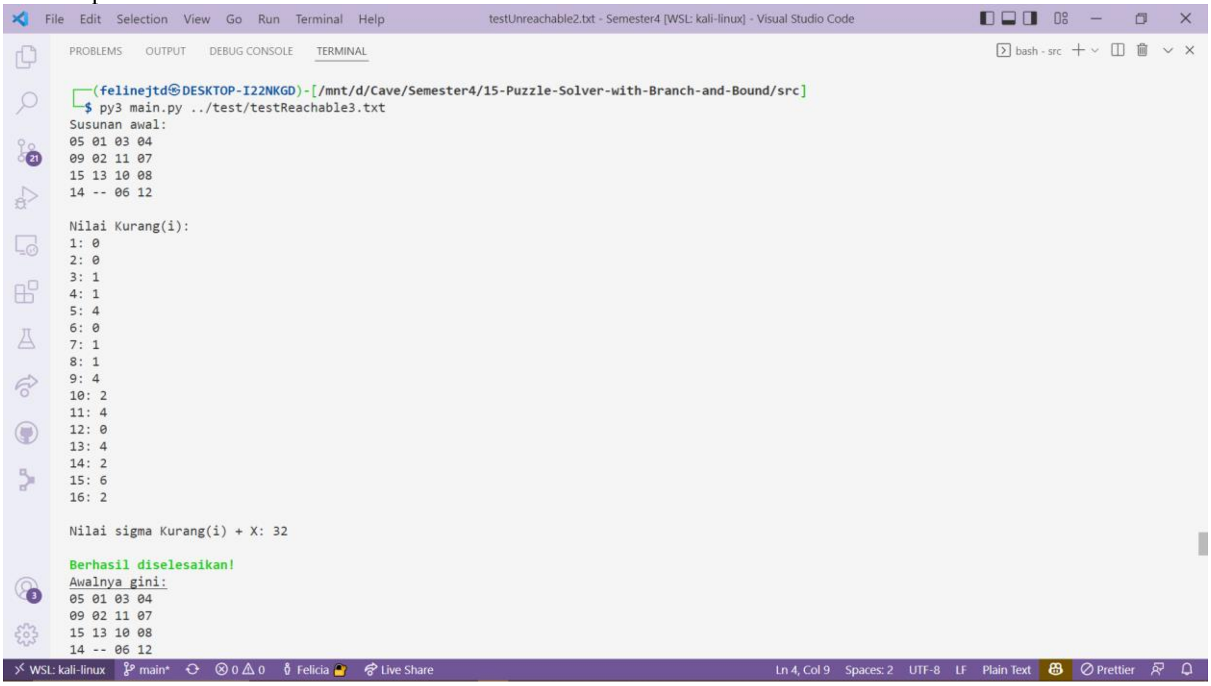
3. Test Case Reachable 3

a. Input

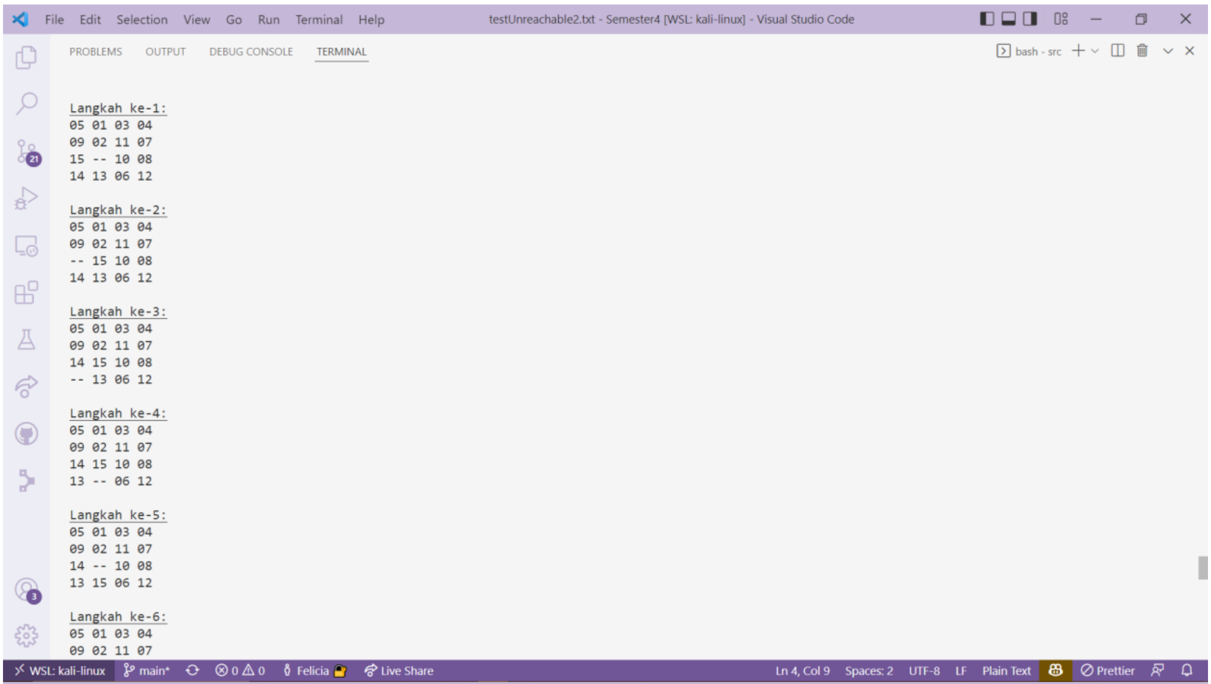

File Edit Selection View ···
testReachable3.txt - Semester4 [WSL: ...
testReachable3.txt
15-Puzzle-Solver-with-Branch-and-Bound > test > testReachable3.txt
WSL: kali-linux
main*
0 0
Felicia
Live Share
Plain Text
Prettier

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

b. Output

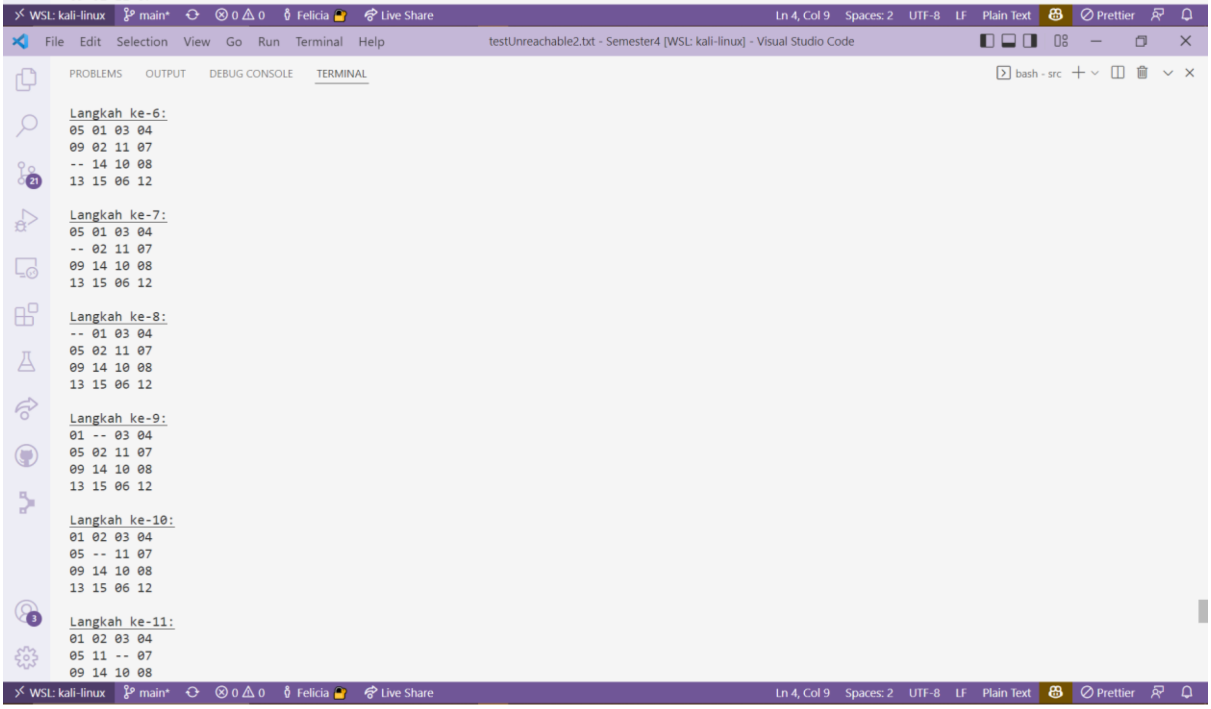

File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux
main*
0 0
Felicia
Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text
Prettier

```
┌──(felinejtd㉿DESKTOP-I22NKGD)-[/mnt/d/Cave/Semester4/15-Puzzle-Solver-with-Branch-and-Bound/src]
└─$ py3 main.py ../test/testReachable3.txt
Susunan awal:
05 01 03 04
09 02 11 07
15 13 10 08
14 -- 06 12

Nilai Kurang(i):
1: 0
2: 0
3: 1
4: 1
5: 4
6: 0
7: 1
8: 1
9: 4
10: 2
11: 4
12: 0
13: 4
14: 2
15: 6
16: 2

Nilai sigma Kurang(i) + X: 32

Berhasil diselesaikan!
Awalnya gini:
05 01 03 04
09 02 11 07
15 13 10 08
14 -- 06 12
```



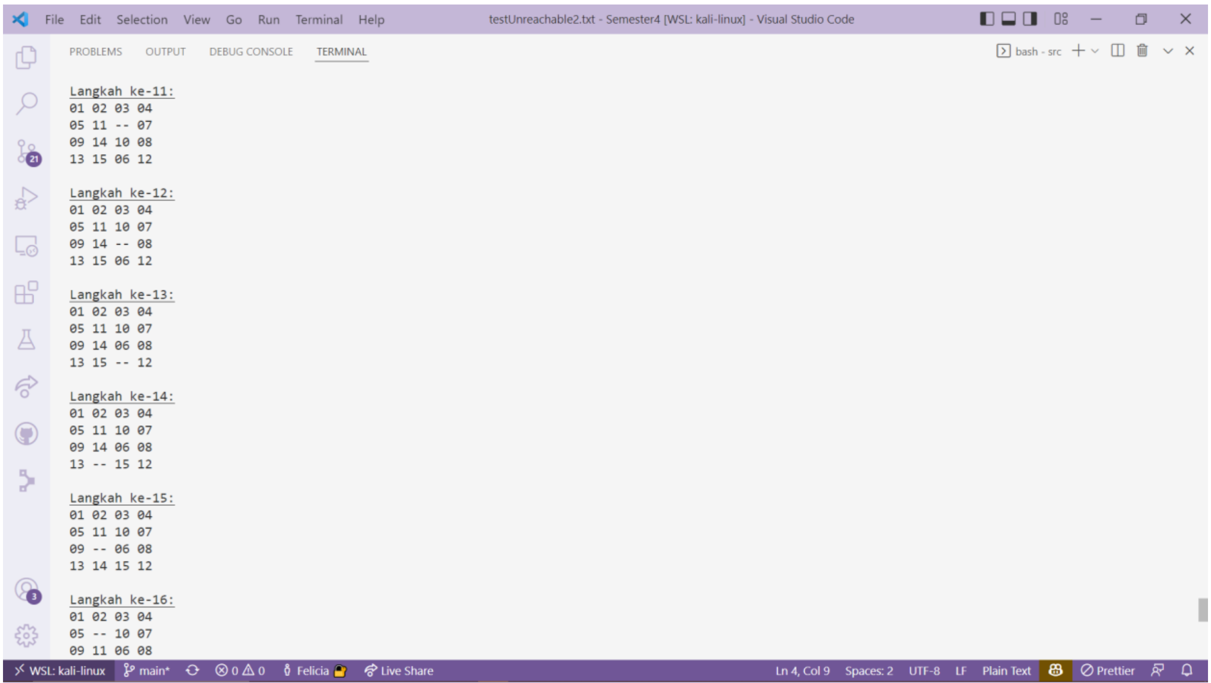
File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux main* 0 0 Felicia Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text Prettier

```
Langkah ke-1:
05 01 03 04
09 02 11 07
15 -- 10 08
14 13 06 12

Langkah ke-2:
05 01 03 04
09 02 11 07
-- 15 10 08
14 13 06 12

Langkah ke-3:
05 01 03 04
09 02 11 07
14 15 10 08
-- 13 06 12

Langkah ke-4:
05 01 03 04
09 02 11 07
14 15 10 08
13 -- 06 12

Langkah ke-5:
05 01 03 04
09 02 11 07
14 -- 10 08
13 15 06 12

Langkah ke-6:
05 01 03 04
09 02 11 07
```

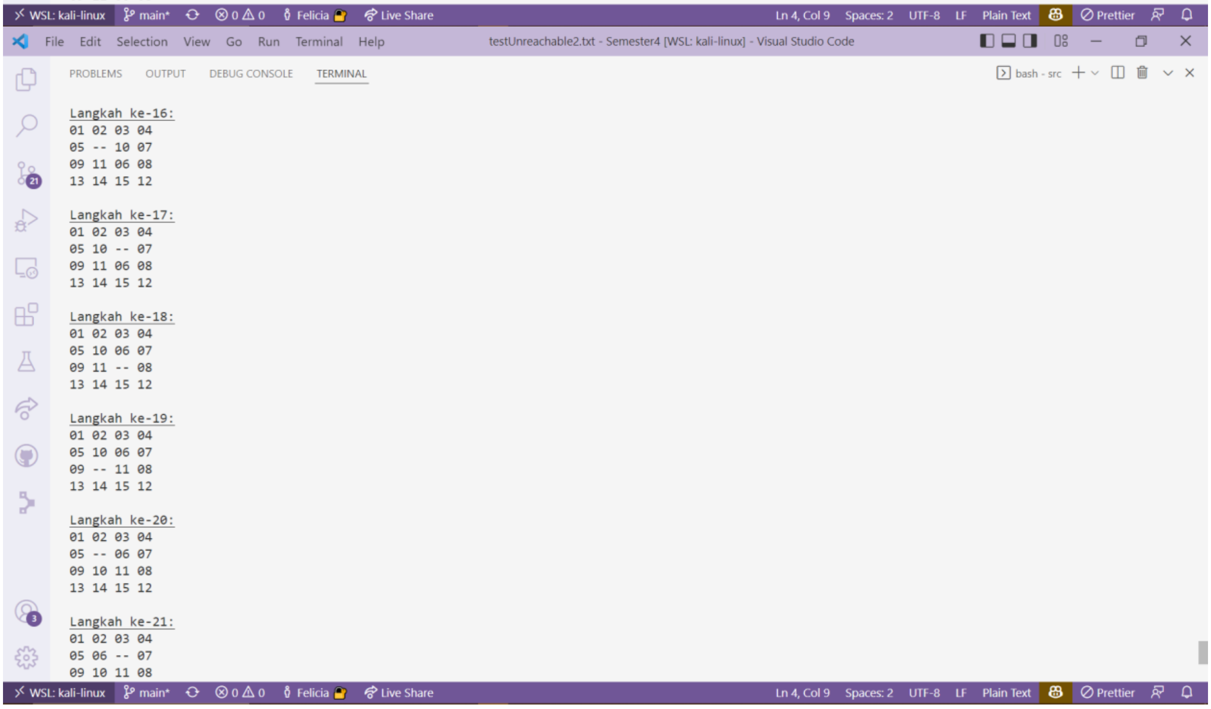


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux main* 0 0 Felicia Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text Prettier

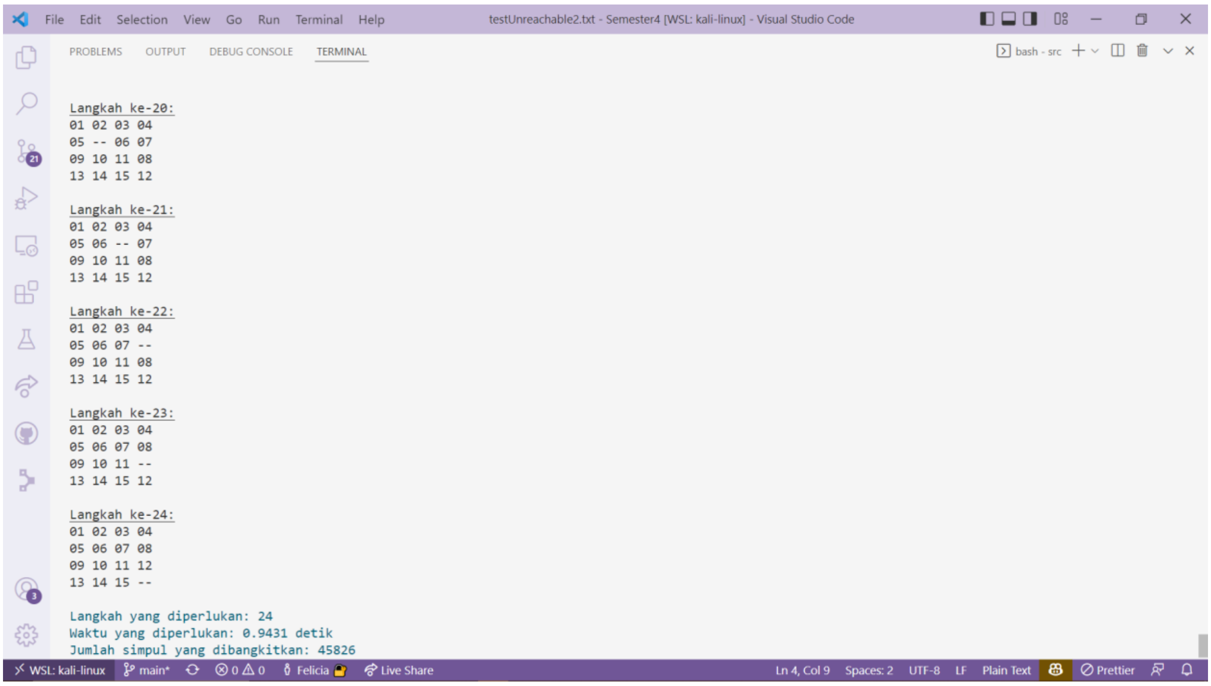
```
Langkah ke-6:
05 01 03 04
09 02 11 07
-- 14 10 08
13 15 06 12

Langkah ke-7:
05 01 03 04
-- 02 11 07
09 14 10 08
13 15 06 12

Langkah ke-8:
-- 01 03 04
05 02 11 07
09 14 10 08
13 15 06 12

Langkah ke-9:
01 -- 03 04
05 02 11 07
09 14 10 08
13 15 06 12

Langkah ke-10:
01 02 03 04
05 -- 11 07
09 14 10 08
13 15 06 12

Langkah ke-11:
01 02 03 04
05 11 -- 07
09 14 10 08
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux main* 0 0 Felicia Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text Prettier

```
Langkah ke-11:
01 02 03 04
05 11 -- 07
09 14 10 08
13 15 06 12

Langkah ke-12:
01 02 03 04
05 11 10 07
09 14 -- 08
13 15 06 12

Langkah ke-13:
01 02 03 04
05 11 10 07
09 14 06 08
13 15 -- 12

Langkah ke-14:
01 02 03 04
05 11 10 07
09 14 06 08
13 -- 15 12

Langkah ke-15:
01 02 03 04
05 11 10 07
09 -- 06 08
13 14 15 12

Langkah ke-16:
01 02 03 04
05 -- 10 07
09 11 06 08
```


File Edit Selection View Go Run Terminal Help
testUnreachable2.txt - Semester4 [WSL: kali-linux] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - src
WSL: kali-linux main* 0 0 Felicia Live Share
Ln 4, Col 9 Spaces: 2 UTF-8 LF Plain Text Prettier

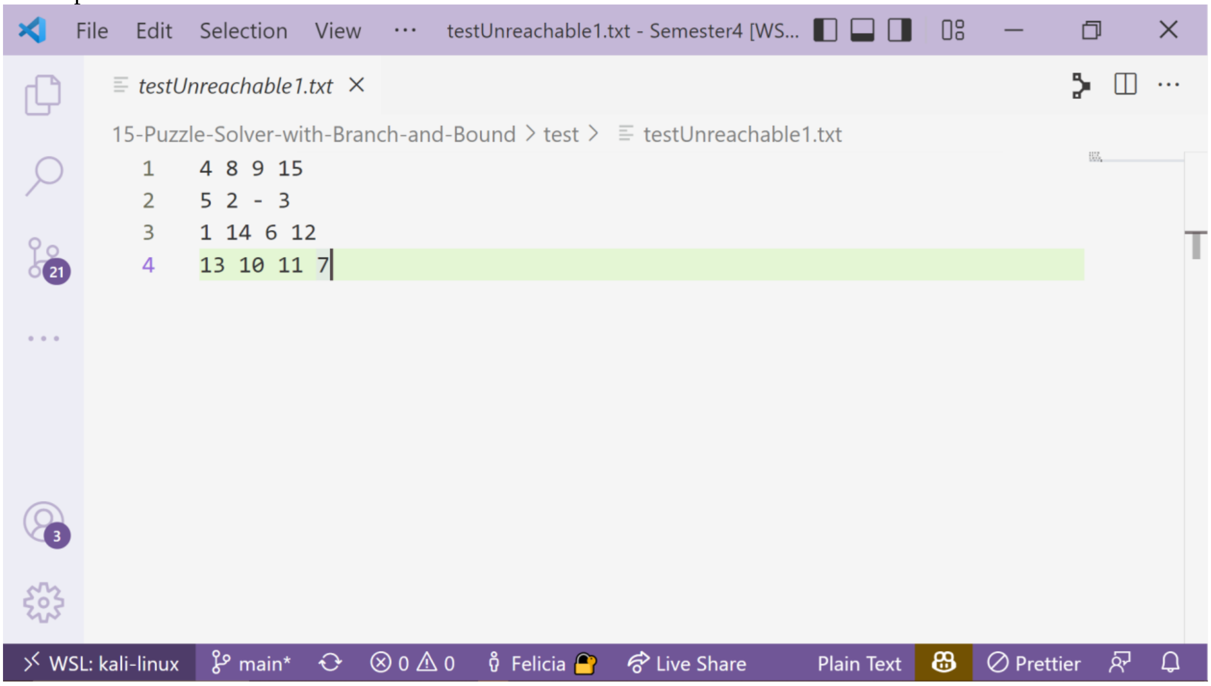
```
Langkah ke-16:
01 02 03 04
05 -- 10 07
09 11 06 08
13 14 15 12

Langkah ke-17:
01 02 03 04
05 10 -- 07
09 11 06 08
13 14 15 12

Langkah ke-18:
01 02 03 04
05 10 06 07
09 11 -- 08
13 14 15 12

Langkah ke-19:
01 02 03 04
05 10 06 07
09 -- 11 08
13 14 15 12

Langkah ke-20:
01 02 03 04
05 -- 06 07
09 10 11 08
13 14 15 12

Langkah ke-21:
01 02 03 04
05 06 -- 07
09 10 11 08
```



```
Langkah ke-20:
01 02 03 04
05 -- 06 07
09 10 11 08
13 14 15 12

Langkah ke-21:
01 02 03 04
05 06 -- 07
09 10 11 08
13 14 15 12

Langkah ke-22:
01 02 03 04
05 06 07 --
09 10 11 08
13 14 15 12

Langkah ke-23:
01 02 03 04
05 06 07 08
09 10 11 --
13 14 15 12

Langkah ke-24:
01 02 03 04
05 06 07 08
09 10 11 12
13 14 15 --

Langkah yang diperlukan: 24
Waktu yang diperlukan: 0.9431 detik
Jumlah simpul yang dibangkitkan: 45826
```

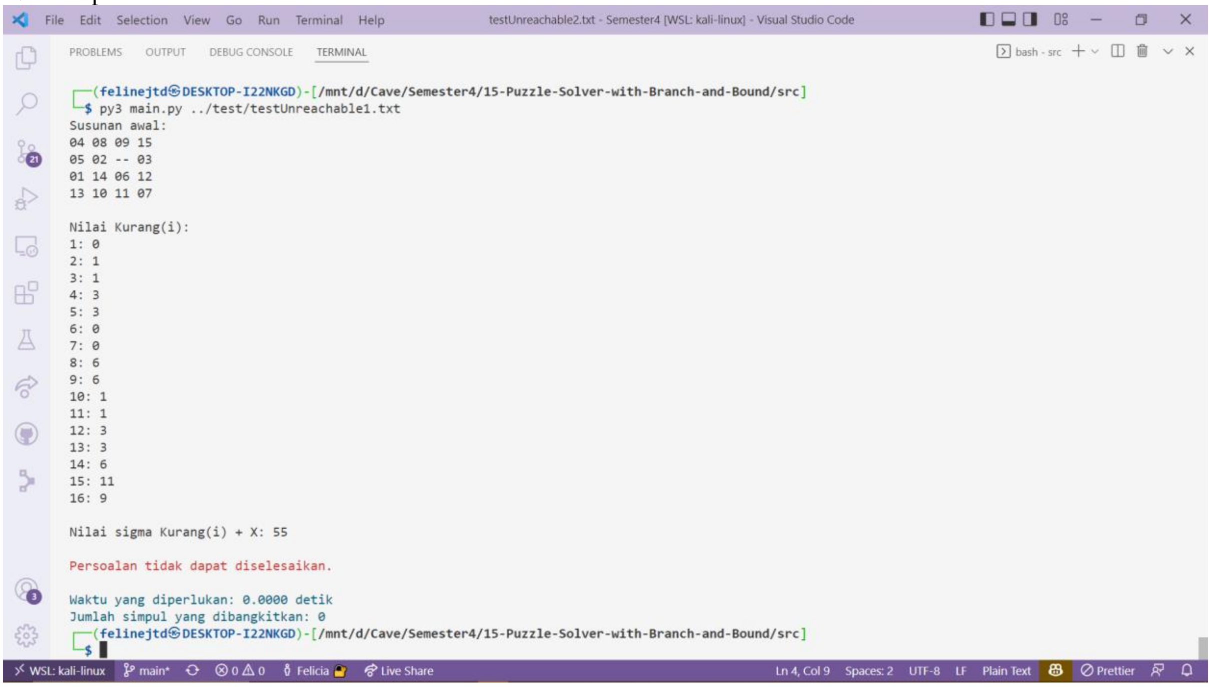
4. Test Case Unreachable 1

a. Input


File
Edit
Selection
View
testUnreachable1.txt - Semester4 [WS...
testUnreachable1.txt
15-Puzzle-Solver-with-Branch-and-Bound > test > testUnreachable1.txt
WSL: kali-linux
main*
0 0
Felicia
Live Share
Plain Text
Prettier

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

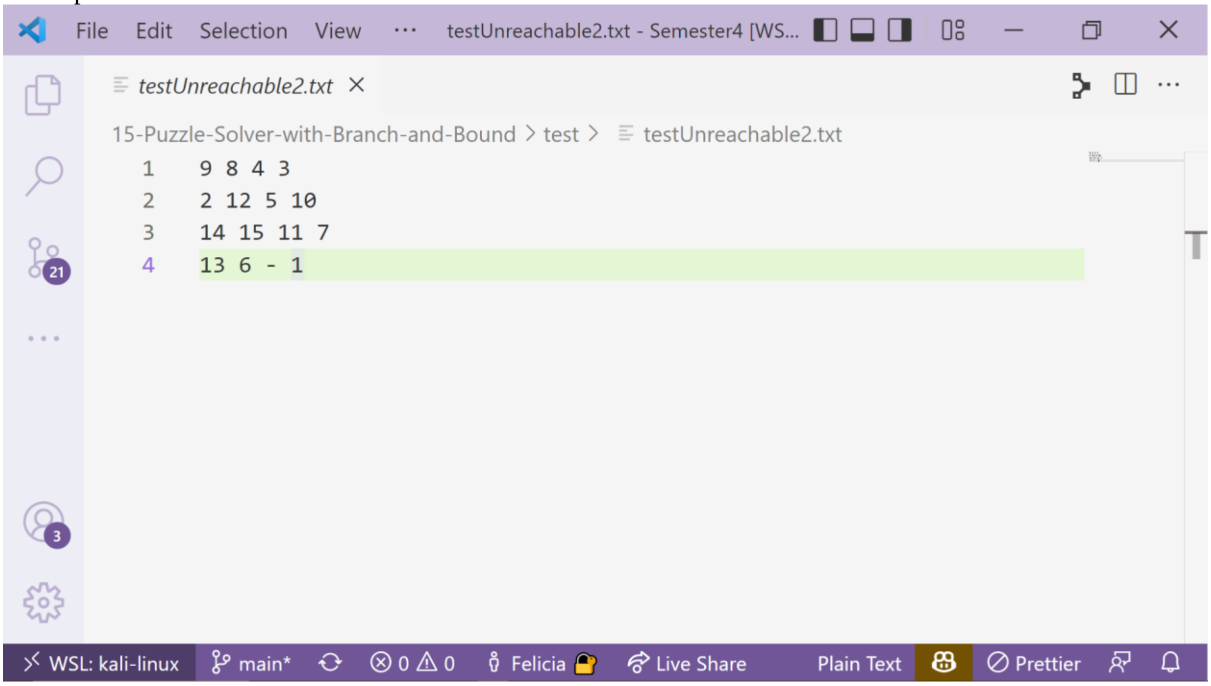
b. Output



```
┌──(felinejtd㉿DESKTOP-I22NKGD)-[/mnt/d/Cave/Semester4/15-Puzzle-Solver-with-Branch-and-Bound/src]
└─$ py3 main.py ../test/testUnreachable1.txt
Susunan awal:
04 08 09 15
05 02 -- 03
01 14 06 12
13 10 11 07

Nilai Kurang(i):
1: 0
2: 1
3: 1
4: 3
5: 3
6: 0
7: 0
8: 6
9: 6
10: 1
11: 1
12: 3
13: 3
14: 6
15: 11
16: 9

Nilai sigma Kurang(i) + X: 55

Persoalan tidak dapat diselesaikan.

Waktu yang diperlukan: 0.0000 detik
Jumlah simpul yang dibangkitkan: 0
┌──(felinejtd㉿DESKTOP-I22NKGD)-[/mnt/d/Cave/Semester4/15-Puzzle-Solver-with-Branch-and-Bound/src]
└─$ ▌
```

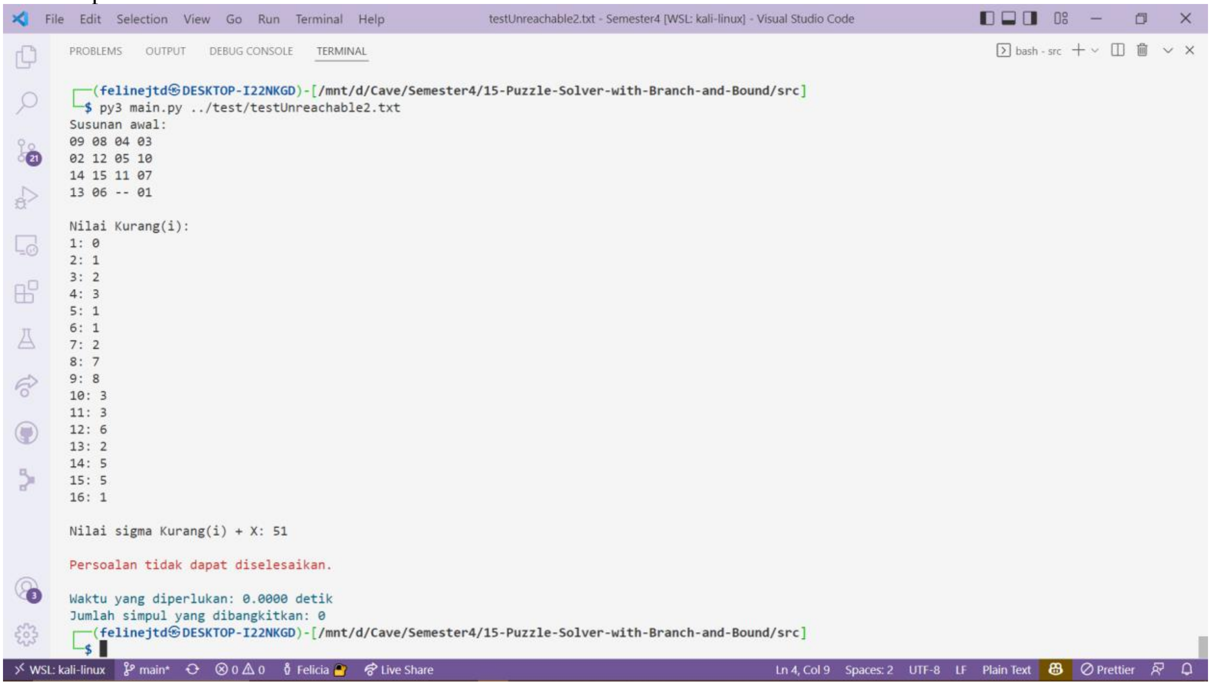
5. Test Case Unreachable 2

a. Input


File
Edit
Selection
View
testUnreachable2.txt - Semester4 [WS...
testUnreachable2.txt
15-Puzzle-Solver-with-Branch-and-Bound > test > testUnreachable2.txt
WSL: kali-linux
main*
0 0
Felicia
Live Share
Plain Text
Prettier

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

b. Output



```
┌──(felinejtd㉿DESKTOP-I22NKGD)-[/mnt/d/Cave/Semester4/15-Puzzle-Solver-with-Branch-and-Bound/src]
└─$ py3 main.py ../test/testUnreachable2.txt
Susunan awal:
09 08 04 03
02 12 05 10
14 15 11 07
13 06 -- 01

Nilai Kurang(i):
1: 0
2: 1
3: 2
4: 3
5: 1
6: 1
7: 2
8: 7
9: 8
10: 3
11: 3
12: 6
13: 2
14: 5
15: 5
16: 1

Nilai sigma Kurang(i) + X: 51

Persoalan tidak dapat diselesaikan.

Waktu yang diperlukan: 0.0000 detik
Jumlah simpul yang dibangkitkan: 0
┌──(felinejtd㉿DESKTOP-I22NKGD)-[/mnt/d/Cave/Semester4/15-Puzzle-Solver-with-Branch-and-Bound/src]
└─$ █
```

D. Instansiasi 5 buah persoalan 15-puzzle

1. **Test Case Reachable 1**
   6 1 2 4
   7 10 - 3
   5 11 12 8
   9 13 14 15

2. **Test Case Reachable 2**
   1 6 2 4
   5 - 3 8
   9 7 15 11
   13 14 10 12

3. **Test Case Reachable 3**
   5 1 3 4
   9 2 11 7
   15 13 10 8
   14 - 6 12

4. **Test Case Unreachable 1**
   4 8 9 15
   5 2 - 3
   1 14 6 12
   13 10 11 7

5. **Test Case Unreachable 2**
   9 8 4 3
   2 12 5 10
   14 15 11 7
   13 6 - 1

E. Alamat *Drive* Kode Program

Kode program dapat diakses menggunakan link *Google Drive* berikut:
https://drive.google.com/drive/folders/1B2Cj4pSHpo2L3NHrXchhTq49aFV-oODv?usp=sharing

Atau menggunakan *Github*:
https://github.com/FelineJTD/15-Puzzle-Solver-with-Branch-and-Bound

F. Tabel Penilaian

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		√