from heapq import heappush, heappop

import time

# === PART 1: REPRESENTASI STATE ===

GOAL\_STATE = ((1, 2, 3), (4, 5, 6), (7, 8, 0))

def print\_board(state):

for row in state:

print(" ".join(str(cell) for cell in row))

print()

def find\_zero(state):

for i in range(3):

for j in range(3):

if state[i][j] == 0:

return i, j

def generate\_moves(state):

x, y = find\_zero(state)

moves = []

directions = [(-1,0), (1,0), (0,-1), (0,1)]

for dx, dy in directions:

nx, ny = x + dx, y + dy

if 0 <= nx < 3 and 0 <= ny < 3:

new\_state = [list(row) for row in state]

new\_state[x][y], new\_state[nx][ny] = new\_state[nx][ny], new\_state[x][y]

moves.append(tuple(tuple(row) for row in new\_state))

return moves

# === PART 2: GBFS ===

def h\_misplaced(state):

return sum(

state[i][j] != GOAL\_STATE[i][j] and state[i][j] != 0

for i in range(3) for j in range(3)

)

def gbfs(start):

visited = set()

heap = [(h\_misplaced(start), start)]

nodes\_expanded = 0

while heap:

\_, current = heappop(heap)

if current == GOAL\_STATE:

return nodes\_expanded

visited.add(current)

nodes\_expanded += 1

for neighbor in generate\_moves(current):

if neighbor not in visited:

heappush(heap, (h\_misplaced(neighbor), neighbor))

return -1

# === PART 3: A\* ===

def a\_star(start):

visited = set()

heap = [(h\_misplaced(start), 0, start, [])]

nodes\_expanded = 0

while heap:

f, g, current, path = heappop(heap)

if current == GOAL\_STATE:

return path + [current], g, nodes\_expanded

visited.add(current)

nodes\_expanded += 1

for neighbor in generate\_moves(current):

if neighbor not in visited:

heappush(heap, (g + 1 + h\_misplaced(neighbor), g + 1, neighbor, path + [current]))

return [], -1, nodes\_expanded

# === PART 4: UJI COBA DAN PERBANDINGAN ===

initial\_states = [

((1, 2, 3), (4, 0, 6), (7, 5, 8)),

((1, 2, 3), (5, 0, 6), (4, 7, 8)),

((7, 1, 2), (4, 8, 3), (0, 5, 6))

]

for idx, state in enumerate(initial\_states):

print(f"\n=== Initial Board {idx+1} ===")

print\_board(state)

# Greedy Best First Search

gbfs\_start\_time = time.time()

gbfs\_nodes = gbfs(state)

gbfs\_end\_time = time.time()

gbfs\_elapsed = gbfs\_end\_time - gbfs\_start\_time

# A\* Search

a\_star\_start\_time = time.time()

a\_star\_path, a\_star\_cost, a\_star\_nodes = a\_star(state)

a\_star\_end\_time = time.time()

a\_star\_elapsed = a\_star\_end\_time - a\_star\_start\_time

# Output tertata

print(">>> GBFS Result")

print(f" Nodes Expanded : {gbfs\_nodes}")

print(f" Time Taken : {gbfs\_elapsed:.6f} sec")

print(">>> A\* Result")

print(f" Path Cost : {a\_star\_cost}")

print(f" Nodes Expanded : {a\_star\_nodes}")

print(f" Time Taken : {a\_star\_elapsed:.6f} sec")

HASIL

python -u "c:\laragon\python\.venv\gbfs1.py"

=== Initial Board 1 ===

1 2 3

4 0 6

7 5 8

>>> GBFS Result

Nodes Expanded : 2

Time Taken : 0.000150 sec

>>> A\* Result

Path Cost : 2

Nodes Expanded : 2

Time Taken : 0.000077 sec

=== Initial Board 2 ===

1 2 3

5 0 6

4 7 8

>>> GBFS Result

Nodes Expanded : 4

Time Taken : 0.000119 sec

>>> A\* Result

Path Cost : 4

Nodes Expanded : 4

Time Taken : 0.000385 sec

=== Initial Board 3 ===

7 1 2

4 8 3

0 5 6

>>> GBFS Result

Nodes Expanded : 770

Time Taken : 0.019934 sec

>>> A\* Result

Path Cost : 14

Nodes Expanded : 225

Time Taken : 0.004215 sec

[Done] exited with code=0 in 0.405 seconds