#Hill Climbing

graph = {

'A': [('B', 3), ('C', 4), ('K', 10)],

'B': [('D', 1), ('E', 0)],

'C': [('F', 2), ('G', 3)],

'D': [],

'E': [],

'F': [],

'G': [],

'K': []

}

start = input("Enter the Start Node: ")

goal = input("Enter Goal Node: ")

def hill\_climb(start, goal, graph):

open = []

close = []

open.append(start)

while open:

node = open.pop(0)

if node == goal:

print("Goal found")

close.append(node)

print(close)

return True

close.append(node)

neighbour = graph[node]

for i in neighbour:

if i[0] not in open:

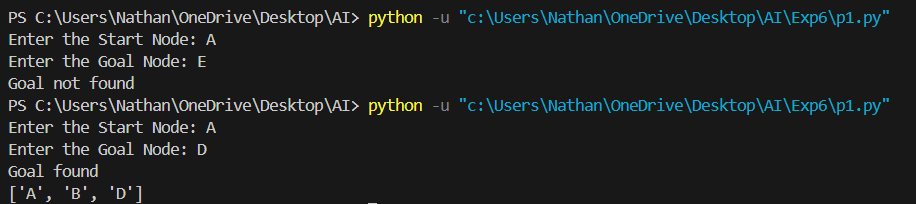
open.append(i[0])

break

print("Goal not found")

hill\_climb(start, goal, graph)

OUTPUT:



# Steepest Ascent Hill Climbing

graph = {

'A': [('B', 3), ('C', 4), ('K', 10)],

'B': [('D', 1), ('E', 0)],

'C': [('F', 2), ('G', 3)],

'D': [],'E': [],'F': [],'G': []

}

heuristics = {

'A': 5, 'B': 3, 'C': 4, 'K': 10, 'D': 1, 'E': 0, 'F': 2, 'G': 3

}

def steepest(start, goal):

current = start

while True:

print("Current Node:", current)

if current == goal:

print("Goal found!")

return

neighbors = graph.get(current, [])

if not neighbors:

print("Stuck at local maxima")

return

neighbors.sort(key=lambda x: heuristics[x[0]])

best\_neighbor, best\_heuristic = neighbors[0]

current = best\_neighbor

start = input("Enter start node: ")

goal = input("Enter goal node: ")

steepest(start, goal)

OUTPUT:  
A screen shot of a computer program

AI-generated content may be incorrect.