Uniform Cost

import heapq

def uniform\_cost\_search(graph, start, goal):

queue = [(0, start, [start])]

visited = set()

while queue:

cost, node, path = heapq.heappop(queue)

if node == goal:

print(f"\nReached goal {goal} with cost {cost}")

print("Path:", " -> ".join(path))

return

if node not in visited:

visited.add(node)

for neighbor, edge\_cost in graph.get(node, []):

if neighbor not in visited:

heapq.heappush(queue, (cost + edge\_cost, neighbor, path + [neighbor]))

print("Goal not reachable.")

# Input graph from user

graph = {}

n = int(input("Enter number of nodes: "))

for \_ in range(n):

node = input("Enter node: ")

neighbors\_input = input(f"Enter neighbors and costs of {node} (format: neighbor1 cost1 neighbor2 cost2 ...): ").split()

neighbors = []

for i in range(0, len(neighbors\_input), 2):

neighbor = neighbors\_input[i]

cost = int(neighbors\_input[i+1])

neighbors.append((neighbor, cost))

graph[node] = neighbors

start\_node = input("Enter start node: ")

goal\_node = input("Enter goal node: ")

print("\nUniform Cost Search Trace:")

uniform\_cost\_search(graph, start\_node, goal\_node)

OUTPUT:

# Example Input:

# Enter number of nodes: 6

# Enter node: A

# Enter neighbors and costs of A (format: neighbor1 cost1 neighbor2 cost2 ...): B 1 C 4

# Enter node: B

# Enter neighbors and costs of B (format: neighbor1 cost1 neighbor2 cost2 ...): D 2 E 5

# Enter node: C

# Enter neighbors and costs of C (format: neighbor1 cost1 neighbor2 cost2 ...): E 1

# Enter node: D

# Enter neighbors and costs of D (format: neighbor1 cost1 neighbor2 cost2 ...): F 3

# Enter node: E

# Enter neighbors and costs of E (format: neighbor1 cost1 neighbor2 cost2 ...): F 1

# Enter node: F

# Enter neighbors and costs of F (format: neighbor1 cost1 neighbor2 cost2 ...):

# Enter start node: A

# Enter goal node: F

# Example Output:

# Reached goal F with cost 5

# Path: A -> B -> D -> F