Greedy Search

import heapq

def greedy\_best\_first\_search(graph, heuristics, start, goal):

queue = [(heuristics[start], start, [start])] # (heuristic, node, path)

visited = set()

while queue:

h\_cost, current, path = heapq.heappop(queue)

if current == goal:

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

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

return

if current not in visited:

visited.add(current)

for neighbor in graph.get(current, []):

if neighbor not in visited:

heapq.heappush(queue, (heuristics[neighbor], neighbor, path + [neighbor]))

print("Goal not reachable.")

# ---------- USER INPUT ----------

graph = {}

heuristics = {}

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

# Input heuristic values

print("\nEnter heuristic values:")

for \_ in range(n):

node = input("Node name: ")

h = int(input(f"Heuristic value of {node}: "))

heuristics[node] = h

# Input adjacency list

print("\nEnter neighbors (unweighted edges):")

for \_ in range(n):

node = input("Enter node: ")

neighbors = input(f"Enter neighbors of {node} (space separated): ").split()

graph[node] = neighbors

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

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

# ---------- RUN GREEDY SEARCH ----------

print("\nGreedy Best-First Search Trace:")

greedy\_best\_first\_search(graph, heuristics, start\_node, goal\_node)

# ---------- SAMPLE INPUT ----------

# Enter number of nodes: 6

#

# Enter heuristic values:

# Node name: A

# Heuristic value of A: 6

# Node name: B

# Heuristic value of B: 4

# Node name: C

# Heuristic value of C: 5

# Node name: D

# Heuristic value of D: 2

# Node name: E

# Heuristic value of E: 1

# Node name: F

# Heuristic value of F: 0

# Enter neighbors (unweighted edges):

# Enter node: A

# Enter neighbors of A (space separated): B C

# Enter node: B

# Enter neighbors of B (space separated): D

# Enter node: C

# Enter neighbors of C (space separated): E

# Enter node: D

# Enter neighbors of D (space separated): F

# Enter node: E

# Enter neighbors of E (space separated): F

# Enter node: F

# Enter neighbors of F (space separated):

# Enter start node: A

# Enter goal node: F

# ---------- OUTPUT ----------

# Greedy Best-First Search Trace:

# Reached goal F

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