*26. A Algorithm*\*

import heapq

def a\_star(graph, start, goal, h):

open\_set = []

heapq.heappush(open\_set, (0 + h[start], 0, start, [start]))

while open\_set:

f, g, node, path = heapq.heappop(open\_set)

if node == goal:

return path, g

for neighbor, cost in graph[node].items():

if neighbor not in path:

heapq.heappush(open\_set, (g + cost + h[neighbor], g + cost, neighbor, path + [neighbor]))

return None, float('inf')

graph = {

'A': {'B': 1, 'C': 4},

'B': {'D': 5},

'C': {'D': 1},

'D': {}

}

heuristic = {'A': 7, 'B': 6, 'C': 2, 'D': 0}

path, cost = a\_star(graph, 'A', 'D', heuristic)

print("A\* Path:", path)

print("Total Cost:", cost)

OUTPUT:

