**from** queue **import** PriorityQueue  
  
**def** ucs(graph, home, guest):  
 **if** home **not in** graph:  
 **raise** TypeError(str(home) + **' not found in graph !'**)  
 **return  
 if** guest **not in** graph:  
 **raise** TypeError(str(guest) + **' not found in graph !'**)  
 **return** *# visited = []* queue = PriorityQueue()  
 queue.put((0, [home]))  
 *# visited.append(home)* **while not** queue.empty():  
 *# print ("Currnet queue is:",queue.queue)  
 # 会取出queue里面cost最小的那个* node = queue.get()  
 *# print ("Node:",node)  
 # 避免重复搜索* visited = node[1]  
 current = node[1][len(node[1]) - 1]  
 *# current = node[1][0]  
 # print ("Current:",current)* **if** guest **in** node[1]:  
 print(**"Path found: "** + str(node[1]) + **", Cost = "** + str(node[0]))  
 **break** cost = node[0]  
 **for** neighbor **in** graph[current]:  
 **if** neighbor **in** visited:  
 **continue** temp = node[1][:]  
 *# print ("Temp:",temp)* temp.append(neighbor)  
 *# print ("Temp append neighbor:",temp)* queue.put((cost + graph[current][neighbor], temp))  
 *# print (queue)***def** main():  
 file = open(**"maps.txt"**,**"r"**)  
 lines = file.readlines()  
 *# 构建一个词典，来保存整个图* graph = {}  
 **for** line **in** lines:  
 *# print (line)* token = line.split()  
 node = token[0]  
 graph[node] = {}  
  
 **for** i **in** range(1, len(token)-1, 2):  
 graph[node][token[i]] = int(token[i + 1])  
 *# graph = retrieval()  
 # print (len(graph["Anyang"]))* print (**"Graph:"**,graph)  
 print(**"Origin:"**)  
 i = input()  
 print(**'\n'**)  
 print(**"Taret:"**)  
 j = input()  
 print (**'The shortest way from '**+ i + **' to '** + j + **':'**)  
 ucs(graph, i, j)  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

运行结果：