Q1：Provide the route planning from Arad to the destination station Bucharest.

Path from Arad to Bucharest: ['Arad', 'Sibiu', 'Fagaras', 'Bucharest']

Q2：Provide the route planning from Oradea and Mehadia to the destination station Bucharest, respectively.

Path from Oradea to Bucharest: ['Oradea', 'Sibiu', 'Fagaras', 'Bucharest']
Path from Mehadia to Bucharest: ['Mehadia', 'Dobreta', 'Craiova', 'Pitesti', 'Bucharest']

Q3：Discuss the Pros and Cons of Greedy Best-First Search.

優點：

1. 速度較快：
   因為Greedy Best-First Search每次的選擇都是目前看起來最有希望的node，所以它可以快速接近目標。尤其是Heuristic Function有效的情況下，能夠顯著縮短搜尋時間。

2. 特定情況效率高：
   在目標比較單一、Heuristic Function準確的問題中，這個算法可能會比其他演算法更有效率，因為它不需要考慮目前路徑的實際成本。

缺點：

1. 不保證最優解：
   如前所述，Greedy Best-First Search只專注於當前看起來最接近目標的節點，沒有考慮實際的路徑成本。因此，可能會找到雖然接近目標但花費更多成本的路徑，不一定會每次都達到最優解。

2. 可能會陷入局部最優解：
   因為它只根據Heuristic Function去選出離目標最近的節點，可能會陷入局部最優解而無法跳出來，導致無法探索其他潛在的更好路徑。

3. 不完整性：
   如果搜索的空間中有死路或是環狀的路徑，Greedy Best-First Search有可能會沒辦法找到任何一條可行的路徑。

Q4：Provide suggestions for improving the disadvantages of Greedy Best-First Search.

1. 對於不保證最優解：
   使用A*演算法，因為A*會同時考慮到實際路徑的成本和估計目標距離，進而更好地平衡最短路徑及目標導向，即可保證找到最優解。

2. 對於可能會陷入局部最優解:
   結合多個Heuristic Function, 進行加權平均或是動態調整權重, 即可避免因為依賴單一的Heuristic Function而陷入局部最優的狀況。

3. 對於不完整性:
   a. 紀錄已經訪問過的Node, 這樣如果同一個Node再次出現在擴展過程時, 就可以直接忽略, 避免重複搜索。
   b. 當發現當前擴展的節點沒有任何有效的下一步選擇, 就回溯到上一個節點, 嘗試擴展其他還未探索的分支。