1. Começamos importando **heapq** para manipular listas e **os** para limpar o terminal;

2. Em seguida, definimos uma classe **Node** para representar os nós do grafo, possui três métodos:
   - __init__ é o construtor, responsável por inicializar os atributos do nó;
   - __eq__ é usado para verificar se dois nós são iguais, retorna true caso verdadeiro;
   - __lt__ é usado para comparar a ordem de dois nós, retorna true se o curto total "**f**" desse nó for menor que o custo total do outro.

3. Prosseguindo, temos uma função que cria e retorna um dicionário representando o grafo do mapa da Romênia com as conexões entre as cidades e as distâncias entre elas;

4. Na próxima parte, definimos o dicionário heurístico, que armazena o valor em linha reta do ponto atual ao ponto de destino;

5. Prosseguindo, implementamos a função **a_star**, que recebe o grafo, nó inicial e nó de destino como argumentos, seu funcionamento passo a passo acontece da seguinte forma:
   - Criamos os objetos do nó inicial e de destino;
   - Inicializamos a lista aberta, para armazenar nós ainda não explorados. E a fechada, para armazenar nós já explorados;
   - Adicionamos o nó inicial a lista aberta;
   - Inicializa o loop principal, que só irá terminar quando open_list estiver vazia;
   - Atribui o nó com o menor custo "**f**" da **open_list** para current_node;
   - Adiciona **current_node** a lista dos explorados;
   - Entramos no bloco que verifica se o **current_node** é o nó de destino, caso sim, inicializamos a lista **path**, entramos em um while, enquanto **current_node** não for **None**, ele adiciona o nó atual a path, e reatribui o nó pai desse nó como **current_node**, retornando dessa forma o caminho do início ao fim;
   - Inicializa uma lista para armazenas os nós filhos de **current_node**;
   - Agora vamos iterar sobre os vizinhos do **current_node** e suas distâncias;
   - Cria um nó filho de **current_node** passando seu vizinho e distância até ele;
   - Aqui calculamos o **g(n)**, somamos o custo do nó atual ao custo da distância que leva ao nó filho;
   - Prosseguindo, calculamos o **h(n)**, chamamos nossa função de custo heurístico para retornar a heurística;
   - Agora calculamos **f(n)**, somando o resultado das duas operações anteriores;
   - Por fim, adicionamos o nó filho à lista de nós filhos;
   - Agora vamos iterar sobre os filhos na lista de filhos para tratá-los de forma adequada, verificamos então se o nó filho atual já está lista fechada, se sim, o **continue** passa para o próximo nó filho;
   - Verificamos se o nó filho já está na lista de nós abertos e se o custo total é maior que o custo total do nó já presente na **open_list**, o loop passa para o próximo filho;
   - Caso o nó filho não atenda as condições anteriores, o nó filho é adicionado a **open_list** para ser explorado;
   - Por fim, os nós serão iterados dessa forma até o caminho ser retornado pelo nosso bloco "**if current_node == goal_node:**", ou o nó destino não ser encontrado e retornarmos **None**;

6. O restante do código está relacionado ao exemplo de uso, definimos o grafo **grafh**, **start_point**, **goal_point**, chamamos a função **a_star** e manipulamos o terminal para printar as informações.