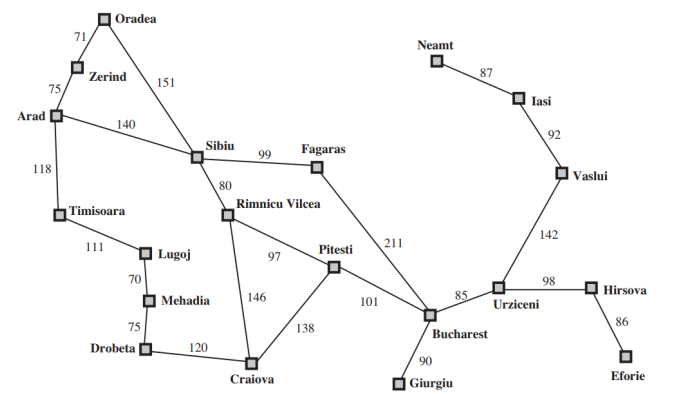
**Question 1. Suppose two friends live in different cities on a map, such as the Romania map shown in below Figure**

****

**On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance d(i, j) between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.**

**a. Write a detailed formulation for this search problem.**

Denote the cities as *C*1​,*C*2​,…,*Cn*​. We can represent the state of the problem as the pair of current cities for the two friends, *S*=(*Ci*​,*Cj*​), where i and j are the current cities of the two friends. The goal is to find a sequence of states that minimizes the total time for the two friends to meet.

The actions in this problem are moving each friend to a neighboring city, and the cost of an action is the road distance between the current city and the neighboring city. The transition model is deterministic since the road distances are known.

The initial state is (*C*start1​,*C*start2​), where *C*start1​ and *C*start2​ are the initial cities of the two friends.

The goal state is (*C*goal​,*C*goal​), where *C*goal​ is the city where the two friends meet.

**State Space**: The state space consists of all possible combinations of cities where each friend is located.

**Initial State**: The initial state is the current locations of the two friends.

**Goal State**: The goal state is achieved when both friends are at the same city.

**Successor Function**: The successor function generates all possible combinations of moving each friend to a neighboring city.

**Cost Function**: The cost of moving from one city to another is the road distance between them.

**b. Let D(i, j) be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible?**

**• D(i, j)**

**• 2 · D(i, j)**

**• D(i, j)/2**

Among these, *D*(*i*,*j*), *D*(*i*,*j*)/2 is admissible because *D*(*i*,*j*) calculates the straight-line distance between the current state and the goal state, which is a lower bound on the actual road distance. Furthermore, *D*(*i*,*j*)/2 is less or equal to *D*(*i*,*j*)

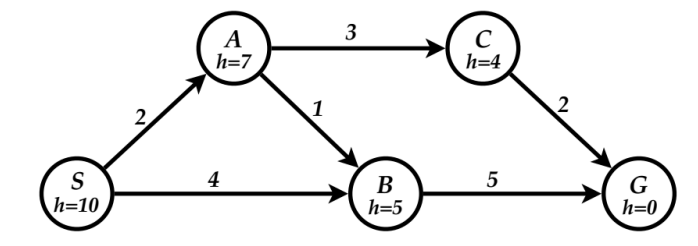
**c. Are there completely connected maps for which no solution exists?**

In a completely connected map, where every city is directly connected to every other city, the solution always exists. As long as there is a road connecting every pair of cities, the friends can eventually reach each other.

**d. Are there any maps in which all solutions require one friend to visit the same city twice?**

If there are cities with loops or multiple paths leading to the same city, it's possible that a solution requires one friend to visit the same city twice. This situation may occur if the road distances are such that it's more efficient for one friend to backtrack and meet the other friend at a later point.

**Question 2. Consider the following graph, in which S and G are the initial and goal states, respectively. The heuristic values are shown under the vertices’ names, while path costs are shown on every edges.**

****

**For each of the search strategies listed below,**

**(a) list, in order, the states expanded,**

**(b) list, in order, the states included in the found path, and**

**(c) show the final content of the frontier (recall that a state is expanded when it is removed from the frontier)**

**When all else is equal, nodes should be expanded in alphabetical order.**

. Uniform-cost search (UCS)

List of expanded nodes: [S, A, B, C, G]

Path found: S, A, C, G

Frontier = { }

. Depth-first search (DFS) (Avoid loops by remembering nodes on the current path).

List of expanded nodes: [S, A, B, G]

Path found: S, A, B, G

. Iterative deepening search (IDS)

List of expanded nodes for each limit: {S} {S, A, B} {S, A, B, C, G}

Path found: S -> B -> G

Greedy best first search (GBFS)

* List of expanded nodes: { S, B, G }
* Path found: S -> B -> G
* Frontier: { A (h=7) }

A\* search

* List of expanded nodes: { S, A, B, C, G }
* Path found: S -> A -> C -> G
* Frontier: { }

From S to G: The actual cost is 9 (S -> B -> G), and the heuristic value is 10. The heuristic is admissible for S.

From A to G: The actual cost is 6 (A -> B -> G), and the heuristic value is 7. The heuristic is admissible for A.

From B to G: The actual cost is 5 (B -> G), and the heuristic value is 5. The heuristic is admissible for B.

From C to G: The actual cost is 2 (C -> G), and the heuristic value is 4. The heuristic is admissible for C.

From G to G: The actual cost is 0 (already at the goal), and the heuristic value is 0. The heuristic is admissible for G.

Since the heuristic values are less than or equal to the actual costs for all states, the given heuristic is admissible.

A heuristic is consistent if the estimated cost from the current state to a successor, plus the estimated cost from the successor to the goal, is less than or equal to the estimated cost from the current state to the goal.

Mathematically: *h*(*n*)≤*c*(*n*,*a*,*n*′)+*h*(*n*′) for every state *n*, action *a*, and successor state *n*′.

1. S to A: h(S) = 10, c(S, S->A, A) = 2, h(A) = 7. It holds (10 <= 2 + 7).
2. S to B: h(S) = 10, c(S, S->B, B) = 4, h(B) = 5. It holds (10 <= 4 + 5).
3. A to B: h(A) = 7, c(A, A->B, B) = 1, h(B) = 5. It holds (7 <= 1 + 5).
4. A to C: h(A) = 7, c(A, A->C, C) = 3, h(C) = 4. It holds (7 <= 3 + 4).
5. B to G: h(B) = 5, c(B, B->G, G) = 5, h(G) = 0. It holds (5 <= 5 + 0).
6. C to G: h(C) = 4, c(C, C->G, G) = 2, h(G) = 0. It holds (4 <= 2 + 0).

The heuristic is consistent.